

TE  
662  
.A3  
NO.  
FHWA-  
RD-  
77-  
123

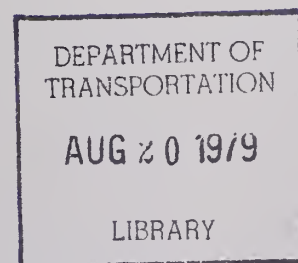
Report No. FHWA-RD-77-123

# VEHICLE DETECTION PHASE III: PASSIVE BUS DETECTOR/INTERSECTION PRIORITY SYSTEM DEVELOPMENT

Option II: Manufacturing Drawings and Prototype Development



October 1977  
Classifier Handbook



Document is available to the public through  
the National Technical Information Service,  
Springfield, Virginia 22161

Prepared for  
FEDERAL HIGHWAY ADMINISTRATION  
Offices of Research & Development  
Washington, D. C. 20590

## FOREWORD

This report is the result of a contract with Honeywell, Inc. to develop a Passive Bus Detector and Intersection Priority system. The system functions as a fully independent traffic controller with completely passive bus detection capability (i.e. with no special equipment required on the bus) using inductive loop detectors (ILD) placed in the roadway.

Sufficient copies of the report are being distributed to provide a minimum of one copy to each Regional Office, and one copy to each Division Office and two copies to each State highway agency. Direct distribution is being made to the Division Offices.

Limited copies for official use are available upon request from Mr. Lyle Saxton, Chief, Systems Development and Technology Group, HRS-32, Washington, D. C. 20590. Additional copies for the public will be available from the National Technical Information Service (NTIS) Department of Commerce, 5285 Port Royal Road, Springfield, Virginia 22161. A small charge will be imposed for each copy ordered from NTIS.

  
Charles F. Schetty

Director, Office of Research

## NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the object of this report.

1. Report No. FHWA -RD-77-123		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle VEHICLE DETECTION - PHASE III: PASSIVE BUS DETECTOR/INTERSECTION PRIORITY SYSTEM Option II: Mfg. Drawings and Prototype Development				5. Report Date October 1977	
				6. Performing Organization Code	
7. Author(s) Rodney M. Larson				8. Performing Organization Report No. F2186-FR4	
9. Performing Organization Name and Address Honeywell Inc. Defense Systems Division 2600 Ridgway Parkway Minneapolis, Minnesota 55413		DEPARTMENT OF TRANSPORTATION  AUG 20 1979  LIBRARY		10. Work Unit No. (TRAIS)	
12. Sponsoring Agency Name and Address Federal Highway Administration Office of Research and Development Washington, D. C. 20590				11. Contract or Grant No. DOT-FH-11-8149	
				13. Type of Report and Period Covered Classifier Handbook	
				14. Sponsoring Agency Code T0242	
15. Supplementary Notes  Contract Manager: Frank J. Mammano					
16. Abstract This handbook is directed towards the engineers who will be adapting the Passive Bus Detector/Classifier to new vehicle identification problems. The handbook begins with a discussion of the passive bus sensor characteristics that are important for vehicle identification. Next, the basic ideas of Pattern Recognition are introduced and discussed from a geometric and trial-and-error point of view. The basic theory of optimal classifier design is presented and compared with the preceding geometric discussion by using the idea data set complexity. These ideas and the operation of the Cascaded Threshold Classifier program are illustrated with two detailed numerical examples. The final chapter of the handbook gives a step-by-step description of the design of the original bus classifier, showing the reasoning and method used in developing the classifier equations. This handbook is written with the idea that optimal methods work well for optimal problems, but practical problems need practical solutions. The Cascaded Threshold Classifier program attempts to meet this practical need. It contains a variety of data manipulation subroutines, including optimal classifier solutions, along with a control structure that allows the user to apply them interactively and thus discover the characteristics of the data and develop the classifier in small steps. A complete flowchart of the program and a fully annotated Fortran listing are included.					
17. Key Words Pattern recognition, Classifier design, Passive vehicle detection.			18. Distribution Statement Document is available to the public through the National Technical Information Service, Springfield, Virginia 22161		
19. Security Classif. (of this report) UNC LASSIFIED		20. Security Classif. (of this page) UNC LASSIFIED		21. No. of Pages 309	
				22. Price	



## PREFACE

The purpose of this handbook is to give its users basic working knowledge of classifier concepts, acquaint them with the procedures used to design the classifier for the passive bus detector, and instruct them in using the computer program that was used in the bus detector classifier design. The classifier that was initially built into the passive bus detector was designed to recognize the 55 passenger urban transit busses manufactured by Flxible, AMG and GMC prior to 1975 and to reject all other vehicles that use city streets. The vehicles that it rejects include inter-city busses, tour busses and school busses. To be this specific, the classifier uses information derived from the physical structure of the vehicles. This means that the classifier may have to be modified to recognize busses made after 1975 or to recognize other kinds of busses (mini-, articulated, etc.). This handbook contains general examples of classifier design problems and it gives a detailed description of the steps and data involved in designing the original bus classifier.

A "classifier" is a machine that observes objects and events in the world and classifies them by assigning each object or event to one of a number of different classes. The bus detector classifier observes vehicles through a buried loop and classifies them into one of two classes - bus or non-bus. The word "machine" is used to indicate the devices, computers and computer programs that do the observing, data processing and decision making. The term "pattern recognition" is often used to denote the area of knowledge that includes the concepts, theories, methods and tricks that are used to design classifiers. Pattern recognition includes a sizeable body of standard statistical methods that have been studied and used for decades. It also includes a large number of empirical procedures that



are useful in special problems and it uses some purely geometrical concepts to describe and work with data in N-dimensional space. This handbook attempts to avoid the esoteric parts of pattern recognition and stay within the bounds of what can reasonably be expected to be useful in working with the bus detector classifier.

CHAPTER 1 of this handbook discusses the problems involved in identifying busses. It briefly reviews the sensing device and the signal properties that make passive identification possible. This is then used to introduce the basic ideas of pattern recognition as they are used in the bus classifier. CHAPTER 2 presents a more detailed discussion of pattern recognition, what it is, where it comes from, what it can do and how it can be used.

Readers familiar with optimal classifier theory may feel that CHAPTER 2 violates all that is pure, right and wholesome. No apology is offered, because this chapter is intended to stress the value of pictorial methods, trial and error, and simple arithmetic.

CHAPTER 3 is a detailed analysis of a 2-dimensional numerical example. The purpose of this chapter is to show how the geometric concepts discussed in CHAPTER 2 can be applied using relatively simple computations. These computations (which the CTC program performs) are shown in detail. CHAPTER 4 continues the same numerical example and shows the input cards and computer printouts for solving the problem with the CTC program. The example is also extended to 3 dimensions to show how the concepts generalize. Detailed operating instructions for the CTC program are included at the beginning of CHAPTER 4.

CHAPTER 5 contains detailed flowcharts of the CTC program. This chapter is primarily for the user who desires to change the program or adapt it for problems other than the bus detector application. However, the general user may find the summary given in Table 18 to be useful in operating the program.

CHAPTER 6 returns to the bus classifier design problem and discusses the design process from beginning to end. This discussion includes more detail than was presented in the original Passive Bus Detector/Classifier report.

Appendixes A, B and C contain the mathematical details of linear discriminant classifiers and the computations performed by the CTC program and Appendix D contains a fully annotated listing of the CTC program. There are certain to be questions about the program function and operation that were not anticipated in CHAPTERS 1 - 6. The appendixes should supply the answers.

A brief reference list is included at the end of the handbook. This list contains the references mentioned in the text and it lists two recent text books (Meisel, Fu) that are particularly good in presenting the theory and the state-of-the-art of pattern recognition.

I wish to gratefully acknowledge the assistance from Bob Touchberry, Tony Tennis and Dave Brown in preparing and assembling the material contained in this handbook.

## TABLE OF CONTENTS

		Page
CHAPTER 1	THE PASSIVE BUS DETECTOR PROBLEM	1
CHAPTER 2	C LASSIFIERS AND PATTERN REC OGNITION	15
	What is Pattern Recognition?	15
	Some Historical Background	17
	Why Use Classifiers?	19
	How is a Classifier Designed?	21
	Multiclass Classifiers	30
CHAPTER 3	EXAMPLES OF CLASSIFYING MULTIDIMEN - SIONAL DATA	34
	3.1 Introduction	34
	3.2 Sample Data Distributions	34
	3.3 A Three-Class Numerical Example	44
	3.3.1 The Multiple Linear Discriminant Solution	45
	3.3.2 Piecewise Linear Solution Using Subclasses	63
	3.3.3 The Layered Machine Classifier	82
	3.4 Interactive Design of Classifiers	89
	3.5 The CTC Philosophy	91
CHAPTER 4	DATA CARDS AND CONTROL CARDS USED IN THE CTC PROGRAM	93
	Data Card Formats and Contents	95
	Background	95
	"One" Card Format	96
	"Two" Card Format	97
	"Three" Card Format	98
	"Four" Card Format	98
	"Five" Card Format	99
	Two-Card Input Format	100
	Control Card Formats and Contents	101
	"Minus One" Card Format	101
	"Minus Two" Card Format	102
	"Minus Three" Card Format	103
	"Minus Four" Card Format	103
	"Minus Five" Card Format	104
	"Minus Six" Card Format	105
	"Minus Seven" Card Format	106
	"Minus Eight" Card Format	106
	"Minus Ninety-nine" Card Format	108



## TABLE OF CONTENTS (CONCLUDED)

	Page
Test Case 1	108
Test Case 2	140
CHAPTER 5      PROGRAM STRUCTURE	162
Field Delimiters	164
CHAPTER 6      DEVELOPMENT OF A PASSIVE BUS DETECTOR	190
Summary	190
Design of the Moving Vehicle Classifier	198
Design of the Stopped Vehicle Classifier	209
Deciding Whether a Vehicle is Stopped or Moving	215
Testing the Complete Classifier	219
Areas of Potential Concern	226
CHAPTER 7      REFERENCES	234
APPENDIX A      DERIVATION OF LINEAR DISCRIMINANT CLASSIFIER FOR THE 2-CLASS UG CASE	235
APPENDIX B      PIECEWISE LINEAR DISCRIMINANTS	246
APPENDIX C      CTC PROGRAM COMPUTATIONS	253
APPENDIX D      LISTING OF THE CTC PROGRAM	261

## LIST OF ILLUSTRATIONS

Figure		Page
1	Waveform produced by a 1972 Flxible 40-foot bus	2
2	Typical Vehicle Waveforms	3
3	Figures Illustrating the Discussion of the Bus Classifier Concept	7
4	Major Function Flow of Bus Classifier	13
5a	Example of Low Complexity Distribution	37
5b	Low Intra-class Complexity with High Inter-class Complexity	38
5c	High Intra-class Complexity with Low Inter-class Complexity	38
5d	High Inter- and Intra-class Complexity	40
6	Test Case 1 Data Scatter Plot	47
7	Contour Approximation to Distribution Shape	55
8	Intra-class Shape Complexity Plot	70
9	Linear Discriminants for Extended Example	81
10	Hardware Logic Implementation of Piecewise Linear Classifier	87
11	Flow Chart Corresponding to Figure 10	88
12	Example Decks	112
13	Test Case One Data	123
14	Method of Scoring from Boundary Location	123
15	Slopes of Pairwise Boundaries	125
16	Labeled Data Points	125

# LIST OF ILLUSTRATIONS (CONCLUDED)

Figure		Page
17	Karnaugh Map Example	154
18	Six Variable Karnaugh Map	155
19	Plot of Test Case Two Data Projected Onto the Plane of the Means	161
20	Flow Charts	166
21	Installation and Demonstration, Minneapolis-33rd and Johnson St. NE	191
22	Vehicle Signature Examples	192
23	Primary Feature Measurements	197
24	Normalized Signature Average	199
25	Polar Plot of Subclass Means	201
26	Joint Plot of Moving Vehicle Discriminant Values	202
27	Joint Plot of Stopped Vehicle Discriminant Values	212
28	Functional Diagram of the Bus Detector	225
29	Leading Edge of a Vehicle Signature	229
30	Magnitude versus Time	229



# LIST OF TABLES

Table		Page
1	Test Case 1 Data, Numerical (As Given Before Design Process)	46
2	Tradeoff for Feature Number 1	50
3	Tradeoff for Feature Number 2	51
4	Pairwise Classifier Coefficients	58
5	Tradeoff for Pair 1, 2	59
6	Tradeoff for Pair 1, 3	60
7	Tradeoff for Pair 2, 3	61
8	Tradeoff for Feature Number 1	67
9	Tradeoff for Feature Number 2	68
10	Pairwise Classifier Coefficients	72
11	Tradeoff for Pair 1, 2	73
12	Tradeoff for Pair 1, 2	74
13	Tradeoff for Pair 1, 4	75
14	Tradeoff for Pair 2, 3	76
15	Tradeoff for Pair 2, 4	77
16	Tradeoff for Pair 3, 4	78
17	Control and Data Card Functions	109
18	Control Card and Data Card Format Summary	110
19	Parameter and Data Listings	113
20	Tradeoff Listing for Feature Number -1	114
21	Tradeoff Listing for Feature Number -2	115

# LIST OF TABLES (CONTINUED)

Table		Page
22	Tradeoff for Feature Number 1	116
23	Tradeoff for Feature Number 2	117
24	Statistical Analysis Output	118
25	Parameter and Data Listing	127
26	Listing Including the Transformed Features	129
27	Tradeoff for Pair 1, 2	131
28	Tradeoff for Feature Number 1	132
29	Tradeoff for Pair 2, 3	133
30	Tradeoff for Feature Number 2	135
31	Tradeoff for Pair 2, 4	136
32	Tradeoff for Feature Number 3	137
33	Possible Combinations of Three Logical Variables	138
34	Logical Tradeoff for Test Case 1	139
35	Test Case 2 Program Outputs, Mode 1	141
36	Test Case 2 Program Outputs, Mode 2	145
37	Logical Mode Outputs for Test Case 2	149
38	Test Case 2 Data, Projected* Numerical Values	160
39	Table of Variables and Arrays Accessible to the Program Writer Through the Use of Control Cards and Data Cards	165
40	Primary and Secondary Features Available in CTC	205
41	Tradeoff Listing for Bus Classifier Design	207
42	Tradeoff Listing for Bus Classifier Design	210

## LIST OF TABLES (CONCLUDED)

Figure		Page
43	Tradeoff Listing for Bus Classifier Design	213
44	Tradeoff Listing for Bus Classifier Design	216
45	Tradeoff Listing for Stopped Vehicle Classifier	220
46	Tradeoff Listing for Complete Classifier	222
47	Bus Classifier Decision Algorithm	224





## CHAPTER 1

### THE PASSIVE BUS DETECTOR PROBLEM

The passive bus detector senses vehicles with a 6-foot\*by 6-foot\*inductive loop buried in the roadway. The loop is driven by 100 kHz oscillator and the phase of the signal is monitored by the loop electronics. When a vehicle passes over the loop, the metal in the vehicle changes the self inductance of the loop and this causes the phase of the induced signal to change. Since the self inductance depends on the position and amount of metal inside the loop, it changes with time as the vehicle moves over the loop and the phase also changes with time. The bus detector observes the instantaneous phase of the induced signal. This is a waveform that is determined by the kind of vehicle, the path it takes over the loop and the speed at which it is traveling. Figure 1 shows the waveform generated by a 1972 Flxible 40-foot\*bus traveling at a constant speed over the loop. The waveform has three "humps," the first is due to the front axle, the second to the fuel tank and the last to the rear axle and motor.

Figure 2 shows similar waveforms generated by five other vehicles. The dray truck and step van each show two distinct humps caused by the front and rear axles and the station wagon shows a slight dip in the middle. The Torino and the Bug only have one hump because the axles are so close together that they both influence the loop at the same time.

The waveform shapes in these examples are characteristic of the different kinds of vehicles, i.e. busses have three humps, trucks have two and cars have one. A human can quite easily distinguish between busses and other vehicles from the shapes of their waveforms. It would seem then that a machine could do the classifying simply by counting the number of humps in the waveform. Unfortunately, there are other vehicles that also generate three

\* 6 ft. = 1.8 m  
40 ft. = 12.2 m

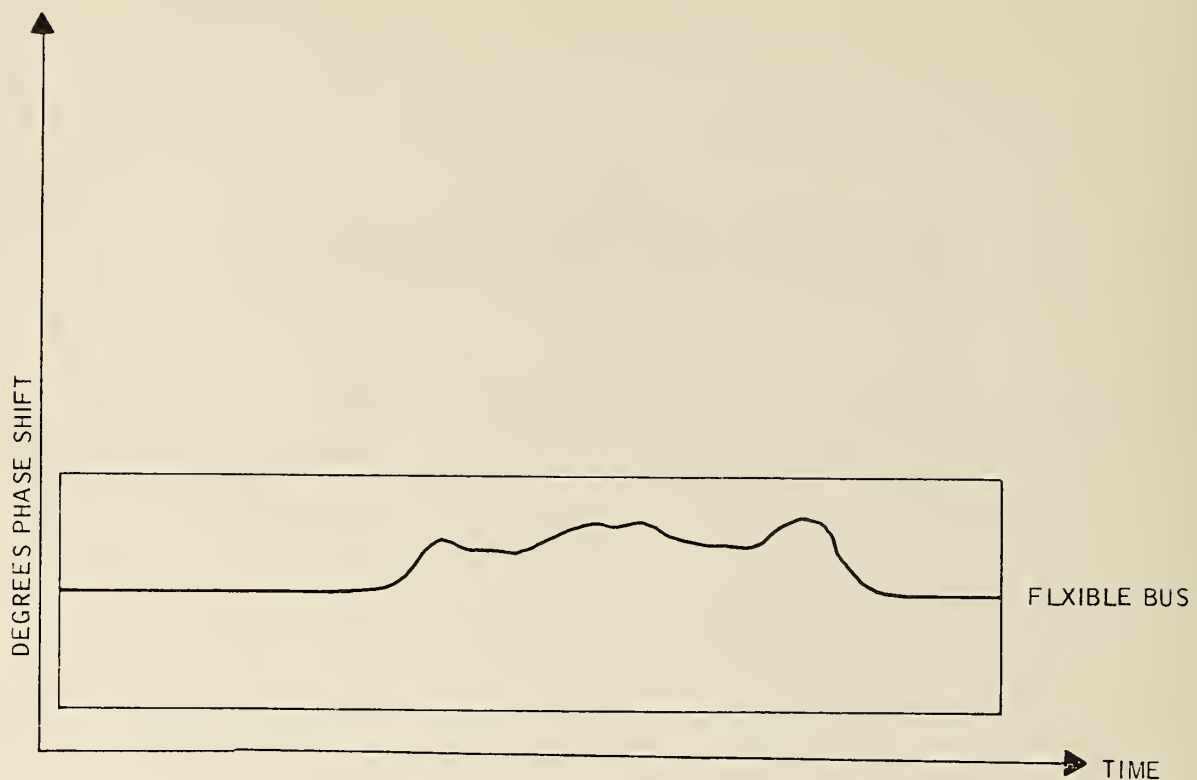


Figure 1. Waveform produced by a 1972 Flxible 40-foot\* bus

(See Volume III, "Vehicle Signature Acquisition," of Final Report on Passive Bus Detector/Intersection Priority System Development, Report No. FHWA-RD-76-66 for detailed information on signal amplitudes and sensor characteristics.)

humps; semitractor trailers have three axles that are far enough apart to show up separately. Thus, three humps is not sufficient to identify a bus. However, the semi has a long distance between the second and third axle while the middle hump of a bus is in the center of the signal. This extra characteristic is almost enough to do the job, but there is a problem with busses that stop or slow down while they are crossing the loop. The change in speed can cause the distance between the second and third humps of a bus signal to be farther apart than normal and cause it to be confused with a semi. Another problem comes from drop-bed moving vans. They generate a hump that is in the middle of the signal and are easily confused with busses. However, if we consider the size of the humps we find that the three humps of busses are almost equal, a semi has a small second hump and a moving van has a large

\* 40 ft. = 12.2 m



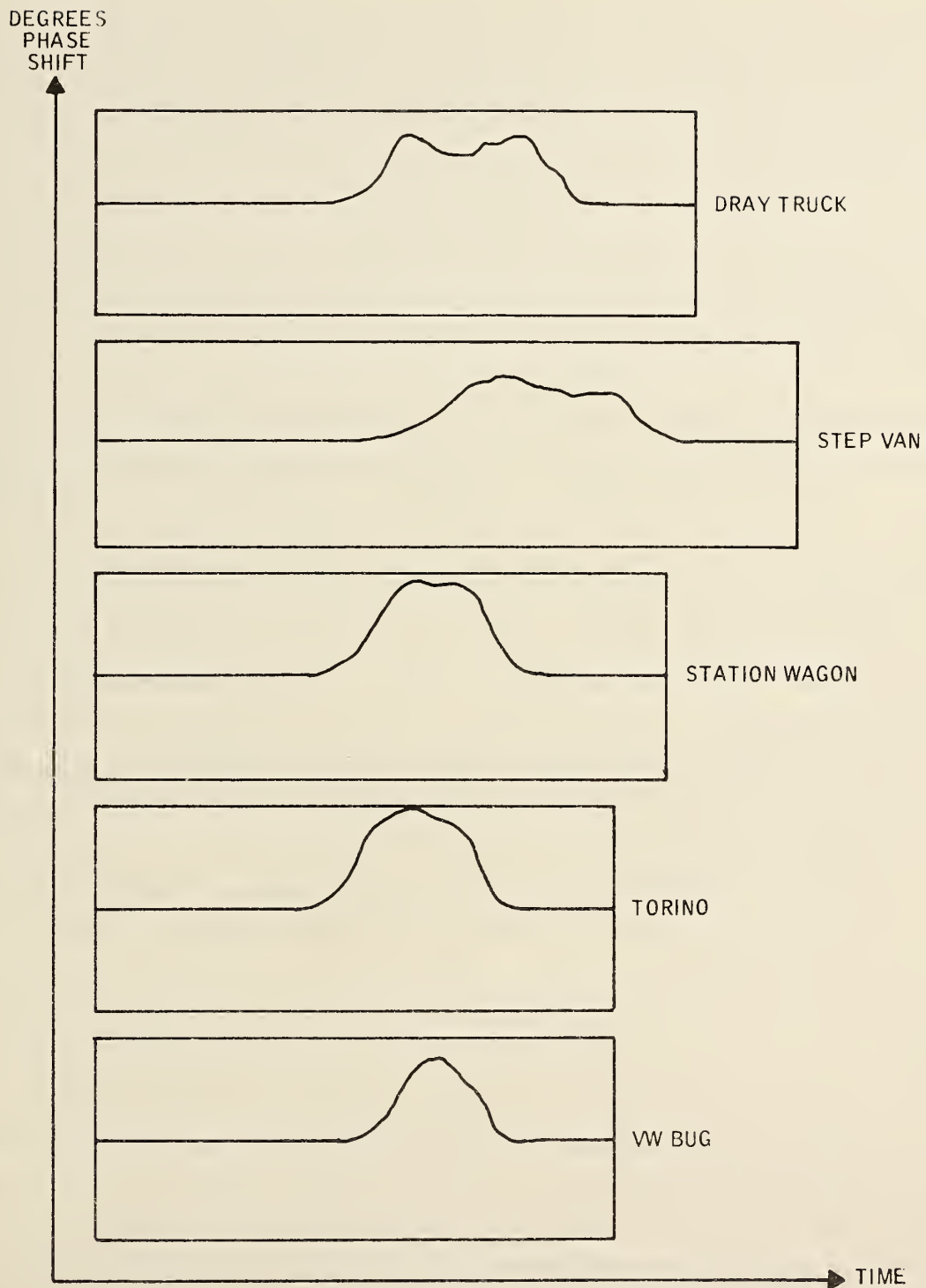


Figure 2. Typical Vehicle Waveforms

second hump. Thus, there is a possibility of classifying by counting the humps and measuring their spacing and amplitude. But, another problem arises when this solution is considered. If we look carefully at the waveforms from the dray truck and the step van in Figure 2, we see that the humps are an illusion. The second hump in the dray truck signal is actually two humps, one small and the other larger. The first hump in the step van signal is really three humps. Thus, we need to give a quantitative definition to what we mean by "hump" since eyesight can be misleading us.

A human with very little training, can rapidly learn to distinguish among shapes like those shown in Figures 1 and 2 such features as spacing, size, what is meant by "hump," and the ways such features combine in different objects, are used automatically. But a machine does not have a human nervous system and it does not have the training that a human has. The machine must start from a state of zero knowledge and zero background and it must be provided with everything it is to use for its job. Thus, in designing a classifier to recognize waveforms, we are led by human perception to choose those features or characteristics that are obvious to us. We then find out how these can be measured or represented to the machine and what rules the machine can use to make the necessary decisions. These steps, (1) define features, (2) measure features, (3) define rules, (4) apply rules to make decision, are involved in designing a classifier. The preceding discussion of the vehicle waveforms shows some of the main considerations involved in a real world problem.

The details of designing the bus detector classifier are discussed later, but a brief discussion of the methods used will help in understanding the more theoretical discussions that follow. The humps that our eyes detect in the waveforms are the main features that were used in the classifier. As mentioned above, the problem in using them is to define precisely what is meant by a hump and to do so in a way that can be tested numerically. The obvious way is to use the local maxima of the waveform, but this has the problem

discussed earlier that there are many small, local maxima that we want to ignore. The solution to this problem was to use a hysteresis filter. This smooths out the small variations and allows the machine to see only the broad, relatively large peaks that we see. The amplitude and spacing of the peaks are then measured. The amplitude is received directly from the loop phase detector electronics and the time interval between successive peaks is read from a digital clock. Since we are measuring broad maxima of the signal, the precise location of the maximum is hard to determine and is sensitive to noise, variations in the vehicle path, and other disturbances. These do not affect the amplitude very much, but they do cause significant statistical variations in the time measurements. The variations make it necessary to use statistical methods in making the decisions.

In general, the rule about three peaks characterizing a bus is true for most of the bus waveforms studied. However, if we look back at Figure 1 we see that the middle peak has two smaller peaks on it. The hysteresis filter will merge these into one peak if they are small enough, but for some busses these peaks are quite large and both are detected. Rather than use a complex rule that sometimes looks for three peaks and sometimes looks for four peaks, the feature extraction method uses only the first, second and last peaks of any signal. This causes the middle peak to move around a little, but this is a relatively small statistical variation.

Another problem that was encountered in the bus detector is that different speeds and different paths over the detector loop cause the time scale and the amplitude scale to change. A fast moving vehicle has a much shorter signal than a slow moving one even though both signals have the same shape. The way this was taken care of was to normalize the amplitudes by using ratios of amplitudes and ratios of time intervals. The features thus used by the classifier are the peak amplitudes relative to the first peak and the time intervals relative to the total time between the first and last peaks. One more problem that bothered the orderly design process was that vehicles do

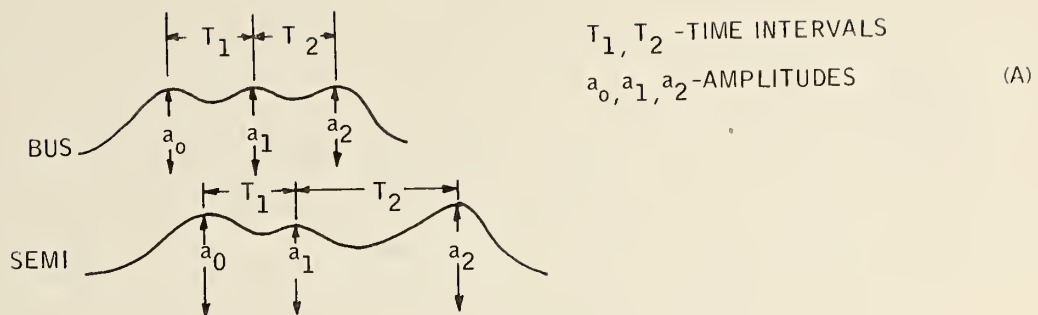


not always keep on moving until they have completely passed over the loop. Sometimes they stop causing the time measurements to increase and the ratios no longer are characteristic of the different vehicle types. This problem was handled by starting the classification by noting how long the vehicle took to cross the loop. (Classification does not start until the vehicle has passed.) If this is large then the classifier assumes that the vehicle has stopped before traveling on and a different decision rule is used than for non-stopping vehicles.

The next important function of the classifier is to decide whether the vehicle that was observed was a bus. A human uses quite subtle and sophisticated methods of recognizing shapes, but the methods are still part of the secret of the human brain. A machine has to use simpler methods based on the best information that it can measure. The preceding discussion leading up to the definition of the normalized amplitude and time interval features was motivated by what the apparent characteristics of the signal were. If the peaks are all of about the same height then the amplitude ratios will all be near unity, and if the second peak is near the middle of the signal then the two ratios will be near one-half. Thus, we should be able to look at the normalized amplitudes and times, compare them to the values one-half and one, and decide that the vehicle was a bus if the features are near these values. Unfortunately, this simple and obvious solution does not work.

The reason that the simple solution fails is the reason that statistical methods are used to find the decision algorithm. A discussion of this will both introduce the ideas behind the decision algorithm and provide motivation for the theoretical discussions to come. Figure 3 illustrates the following discussion. Figure 3a shows the kind of waveforms that we expect from "ideal" busses and semis. The time intervals between peaks and the amplitudes of the peaks are indicated. For an approximately ideal bus we expect  $t_1 = t_2$  and  $a_0 = a_1 = a_2$ . Figure 3b gives definitions for calculating normalized features. These are not the exact definitions used in the bus detector, but they are somewhat





DEFINITION OF NORMALIZED TIME AND AMPLITUDE FEATURES:

$$T_1 = \frac{T_1}{T_1 + T_2}, \quad T_2 = \frac{T_2}{T_1 + T_2} \quad (B)$$

$$A_1 = \frac{a_1}{a_0}, \quad A_2 = \frac{a_2}{a_0}$$

FOR THESE TWO IDEAL WAVE FORMS THE FEATURE VALUES ARE:

	$T_1$	$T_2$	$A_1$	$A_2$
BUS	0.5	0.5	1.0	1.0
SEMI	0.3	0.7	0.7	1.2

(C)

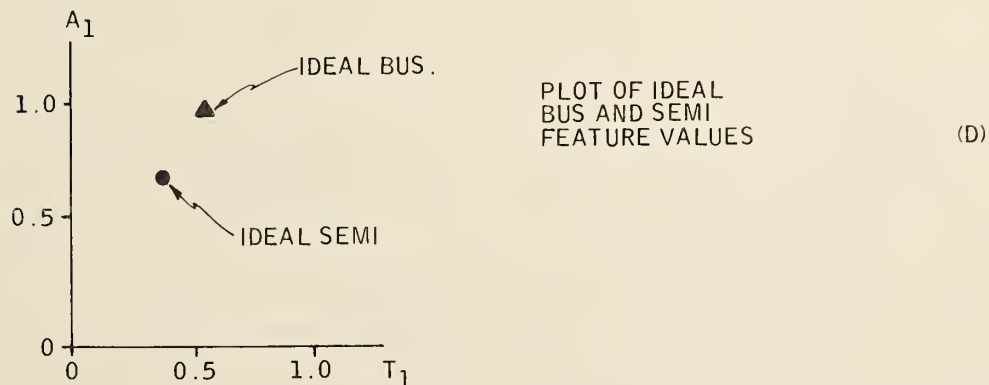
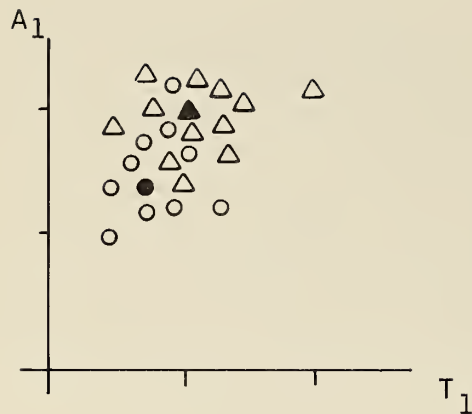


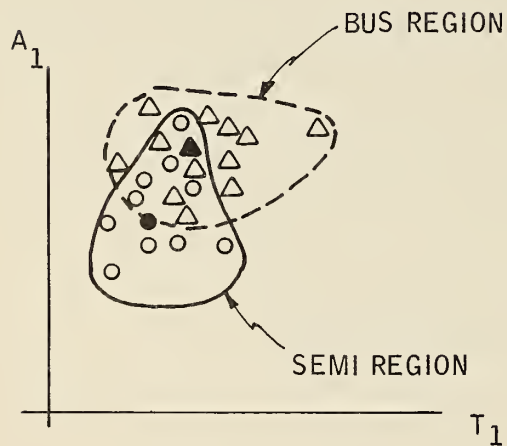
Figure 3. Figures Illustrating the Discussion of the Bus Classifier Concept



TYPICAL PLOT OF  
REAL BUS AND  
SEMI FEATURE VALUES

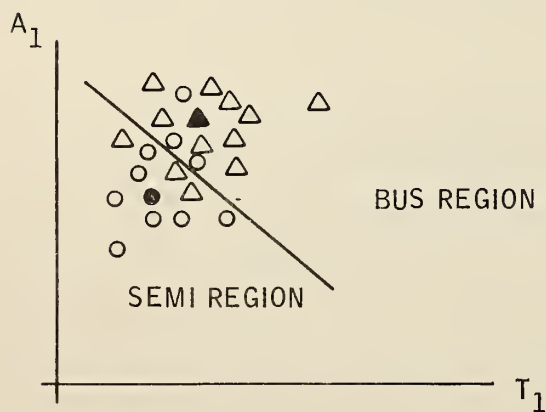
(E)

$\triangle$  - BUS  
 $\circ$  - SEMI



SMOOTH CURVES THAT  
ENCLOSE THE BUS  
AND SEMI SAMPLES

(F)



STRAIGHT LINE SEPARATING  
THE BUS AND SEMI  
DECISION REGIONS

(G)

Figure 3. Figures Illustrating the Discussion of the  
Bus Classifier Concept (Concluded)

simplified for this discussion. Figure 3c lists the expected feature values for the ideal bus and semi waveforms shown in 3a and the values of  $A_1$  and  $T_1$  are plotted in Figure 3d. If all busses were identical to the ideal bus and all semis were identical to the ideal semi, then the classification algorithm would only have to determine which of these two ideal points fit the measurements from a particular vehicle. But real world data is not ideal and Figure 3e shows what a plot of real world data is likely to look like.

The scatter in the points shown is from many sources. Vehicles change speed while they are crossing over the loop and this makes the time intervals differ from the ideal, causing scatter in the  $T_1$  direction. Different makes of semi's and busses are built differently and this causes differences in the amplitude of the peaks which, in turn, causes scatter in the  $A_1$  direction. Noise in the sensing system and inaccuracy in precisely locating the position of the peaks cause scatter in both directions. Because of the scatter, the decision has to be made, not by comparing with a single reference point in feature space, but by comparing with a reference region. Figure 3f shows two regions that describe the distribution of the bus samples and the semi samples. Given an unknown signal, we could calculate the normalized features and plot them on this graph. If the point fell inside the dashed line we could call it a bus and if it fell inside the solid line we could call it a semi. However, there is a part of the plane where the two regions overlap and there is no obviously best decision for points that fall in the overlap region. The theoretically best solution for this case depends on knowing the statistical distribution of the scattering for each class and deriving an optimal boundary from this information. The common sense solution is to draw a line through the middle of the overlap region and call half of it bus and other half semi. For cases where the points can be plotted on graph paper the common sense solution will usually give as good a result as the optimal solution.

There are some difficulties with the solution proposed in the last paragraph. First, suppose that we can discover data clusters that can be described by

closed curves like those shown in Figure 3f. It is quite difficult to represent curves other than circles and straight lines in a computer, (not impossible, just difficult). Second, if we do find a representation of a complicated curve it is usually a long, complex process to determine whether a given point lies inside or outside of it. Third, real world data do not often fall into simple clusters like those shown in Figure 3f, and when they do it is usually a hard problem just to find where the clusters are. The way out of these difficulties is to use a straight line to separate the regions, like that shown in Figure 3g.

A straight line is easy to describe in a computer, it is easy to determine which side of the line a given point lies on and for many types of problems the straight line is the optimal solution. Besides having these virtues, any curve that might be used to divide the regions can be approximated as closely as desired by a number of straight line segments joined end to end. Thus, in practice, straight lines are all that are used.

Before leaving this example let us consider why the regions for different types of objects should overlap and what can be done about it. In every measurement process there is a certain amount of noise that can be defined as the unknown or uncontrolled or unwanted part of the measurement. In a classifier the measurements are used to distinguish between different things and as long as the noise is small compared to the difference that is being tested, then the decision can be made. However, if the noise gets larger than half of the difference between the two way to overcome this problem is to make a number of independent measurements. This can be done in two different ways. If you are measuring a quantity that can be measured more than once, then the different measured values can be averaged and the noise, for independent measurements, will be reduced by the square root of the number of values used. But this will not work for the bus detector because each vehicle only passes the loop once and the decision must be made from that one observation. It is still possible to make independent measurements, but they are measurements of different characteristics of the vehicle. Thus, in the



example in Figure 3, two amplitudes and two time intervals were used. In so far as the noise in each of these measurements is independent from the noise in the other measurements there will be less overlap in the four dimensional space than there is in any of the single measurements <sup>(1)</sup>.

There remains one classifier function that must be discussed - vehicle detection. Before measurements can be made on the vehicle waveform, the machine must know that a vehicle is present and before features can be calculated and used to classify the waveform, the machine must know that the vehicle has passed and all the information has been obtained. In the bus detector, vehicle presence is determined by an amplitude threshold. When the phase shift becomes larger than a specified amount, the machine starts making measurements and when the phase shift then drops below the threshold value the normalized features are calculated and the signal is classified. The threshold value was chosen small enough to detect all of the busses and yet large enough so that it did not trigger on the low level noise output from the phase detector.

---

(1) To see how different independent measurements help reduce overlap, consider the ideal case shown in Figure 3d and suppose that the noise in measuring  $A_1$  is either +0.15 or -0.15 for both vehicle types and the noise for  $T_1$  is either +0.1 or -0.1. Thus if we made two random measurements on busses we would expect to get the two different values  $A_1 = 0.85$  and  $1.15$  and two observations of semi's would give  $A_1 = 0.55$  and  $0.85$ . This is a 50% overlap on this one measurement. Similarly there is a 50% overlap in measurements of  $T_1$ . But if we use both features together, then in four observations of each vehicle type we would expect to get the pairs of values shown:

$$(A_1, T_1) = \begin{array}{cc} (0.85, 0.4) & (0.85, 0.4) \\ (0.85, 0.6) & (0.85, 0.2) \\ (1.15, 0.4) & (0.55, 0.4) \\ (1.15, 0.6) & (0.55, 0.2) \end{array}$$

The overlap is only 25% in this case.



Figure 4 summarizes the functions discussed above and shows how they are connected. The data processing flow begins with the digital output of the phase detector. As each new value of the phase is generated it is compared with the threshold value to determine whether a vehicle is present. If nothing is happening, (small signal) then the machine waits for the next phase value to be generated by the phase detector. If a vehicle is present, (large signal) then the machine notes the starting time and checks each new phase value to see whether it is a signal peak. The times and amplitudes of the peaks are saved. While it is looking for peaks it also compares the phase values with an "end of pass" threshold and if the signal is smaller than this threshold then the ending time is stored and the machine proceeds to the next stage of operation. The first question asked by the machine is "Does the signal have 3 or more peaks?". If not, then the machine declares that the vehicle was not a bus and it goes back to looking for the next vehicle. If there are three or more peaks then the normalized features are calculated from the raw time and amplitude measurements. The next test decides whether the vehicle stopped while it was crossing over the loop. This is done by comparing the total time of vehicle presence with the nominal value of 15 seconds. If the vehicle took more than 15 seconds to cross the loop then the classifier treats it as a stopping vehicle and if less than 15 seconds it assumes the vehicle was moving continuously while crossing the loop. The stopping vehicles are classified by using only the normalized peak amplitudes as was discussed earlier. The moving vehicles are classified by using both the normalized amplitudes and the normalized times. In either case, after announcing the decision, bus or non-bus, the machine then returns to the top of the loop and begins looking for the next vehicle.

The discussion in this chapter has covered all of the main ideas involved in designing a classifier that can distinguish between busses and other kinds of vehicles. The following chapters will treat these same ideas in more general terms. This includes discussion of the statistical and geometric concepts that are used to make sense out of the multi-dimensional data that is used to design

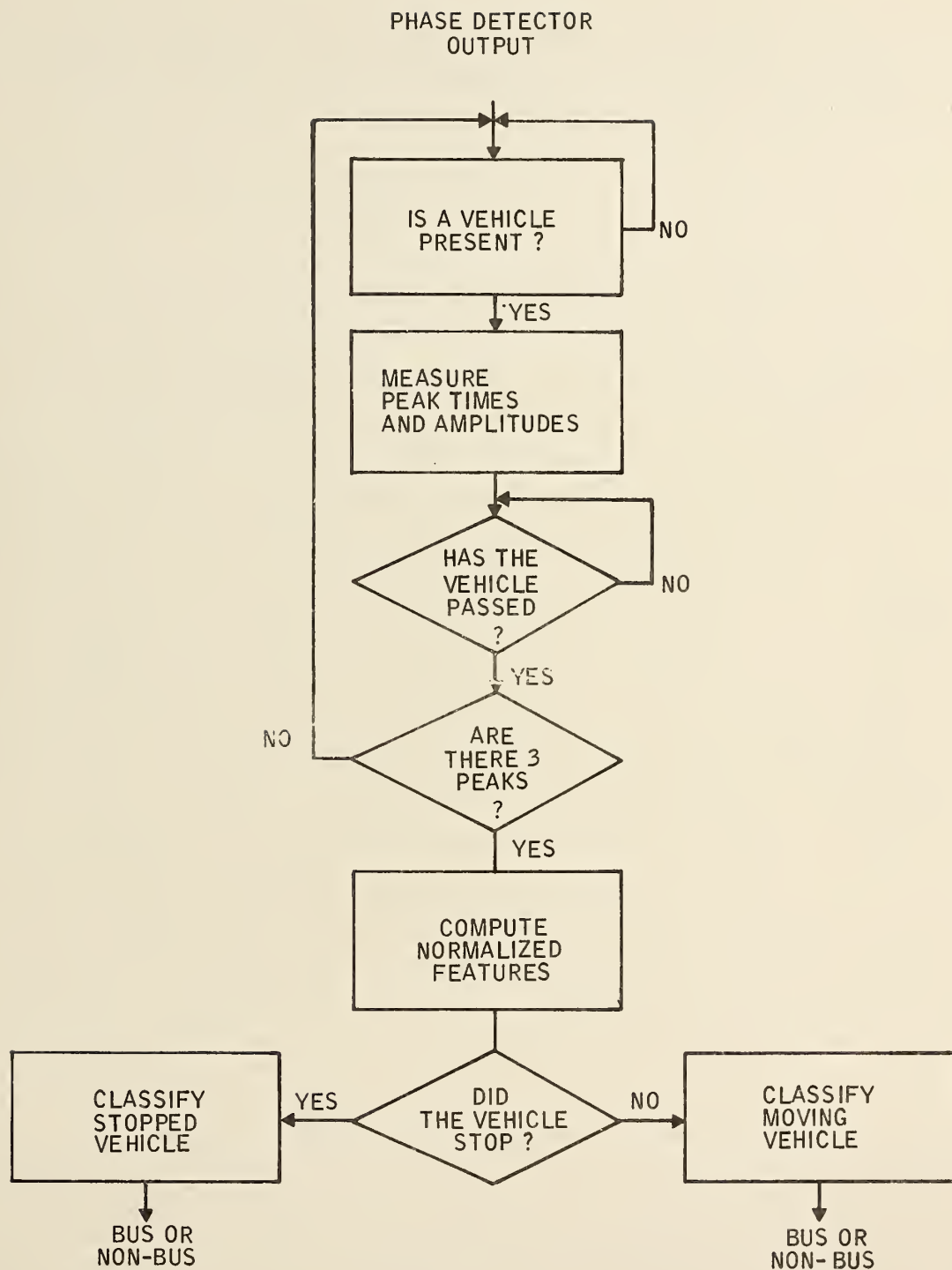


Figure 4. Major Function Flow of Bus Classifier

the classifier. Finally, the computer program that does the number crunching is described. Both the philosophy and the operation of this program are treated with textbook and actual examples.

## CHAPTER 2

### CLASSIFIERS AND PATTERN RECOGNITION

#### WHAT IS PATTERN RECOGNITION?

The idea of 'pattern' comes from human perception. From the time of birth, our senses are bombarded by inputs from the world around us. We learn to sort these inputs into the ones with meaning and the meaningless ones. We further learn to sort the meaningful ones into those we should respond to and those we can ignore. The learning continues to higher and higher levels of sophistication, leading to language, music, mathematics and other subtle thought processes. An important ability that helps make all of this possible is the ability to recognize that something being experienced right now is similar to something we have seen, heard, thought, etc. before. For instance we recognize a truck because we know what a truck looks or sounds or smells like. The things we call 'truck' all have certain properties in common that make them like each other and different from other things - they fit the pattern that we have for trucks. The truck pattern is not something exact, because there are many different kinds of trucks that fit into it. It would be very hard to describe the pattern we use to recognize trucks, because almost any property that a truck has can also be found in something else. (For example: trucks carry things, but so do wagons; trucks have motors, but so do cars; trucks are bigger than cars, but so are busses; etc.)

This may sound very trivial, until you try to build a machine that will recognize trucks automatically. The machine must have a detailed description of the pattern. Pattern recognition theory is concerned with the methods used to discover the detailed pattern descriptions that machines can use to recognize patterns.



It will be worthwhile to dig a little deeper into the ideas in the last paragraph. It is clear that the 'truck pattern' is not itself a truck! Trucks fit the pattern, but the pattern is not a truck. Nor is the pattern a collection of trucks of various kinds! There is an abstract thing called "the set of all trucks" and the things we call trucks are the things in this set, but the set is not the pattern either. The truck pattern is the criteria or descriptions and characteristics that can be used to decide whether a given object is or is not a truck. Pattern recognition is the job of discovering these criteria.

There is another set of terminology used to talk about these matters. The ideas are most easy to discuss using the pattern idea, but to talk about the machines that do the recognizing, it is easier to use the terms 'class, classifier and classification'. A class is the set of all objects that match a certain pattern - like the set of all trucks. A classifier is the machine that decides whether a given object matches the pattern. Classification is work that the object recognizing machine does. In this terminology, pattern recognition is described as "classifier design". Depending on which part of the classifier design problem we are discussing, it will be simpler to use one or the other of these jargons. Thus, it is easier to say "X is in the truck class" than "X matches the pattern called 'truck'", even though the two statements have the same meaning.

A word of caution to those who plan to read some of the references on pattern recognition. Pattern recognition is a relatively new area of technology and different people give different meanings to the terms defined above. Some of the more theoretically minded workers in the field define a pattern as the representation of a particular observation made upon an object (for example, the length, width and color of a given object). Observations always include noise, thus two observations of the same object will yield two different patterns for the object. They are thus lead to defining a "pattern class" as the collection of different patterns that come from different observations of one kind of object. Then "pattern recognition" becomes what we have



called classification. Another group uses the term "pattern" to denote the characteristics by which an object can be recognized. Thus a checkerboard has a square pattern and a backgammon board has a triangular pattern. The term "taxonomy" is sometimes used in place of "classification" by British authors and workers in biology and medicine. It is therefore worthwhile to glance through the preface of a reference book to learn what the terms mean to the author.

## SOME HISTORICAL BACKGROUND

A classifier's job is to make a decision based on some observed characteristics of an object. Before the computer age, all but the simplest decisions had to be made by humans. There were devices like burglar alarms, governors, light switches and locks that could make some very simple decisions, but most machines worked at replacing the human's muscles, not his brain. Then calculators came along to help with arithmetic and they were followed by the sophisticated machines we call computers. The first computers only tried to do more complex mathematics. However it was found that, in order to do the complex jobs, they had to be able to select among alternative courses of action based on the results of the calculations. The next short step was to use this ability to make choices the main job and the mathematics secondary. Thus there are process control computers today that do logical and input/output operations at fantastic speeds, but they don't even have built-in arithmetic.

With such devices available, the next question is how do we use them? What methods, numerical, logical or otherwise, can make a computer perform a complex decision task like recognizing a truck or understanding a spoken word? The answer, of course, is classifiers. Does this mean that all computerized methods of making decisions are called classification? That is exactly what it means! Though it is a young field, pattern recognition theory

has adopted every decision making method it could find, and it has developed new methods where it could not find old ones that could do the job.

The main area that was concerned with decisions before pattern recognition came into being was the branch of statistics called Statistical Hypothesis Testing. It had developed rigorous methods for choosing among alternatives on the basis of random samples and assumptions about their distributions. The random samples are numerical values, either counts or measurements, and the decision is based upon how well the value matches the assumed distributions. Early pattern recognition methods began by representing objects by numerical values called features. These were chosen so as to be statistically different for objects from the different classes of interest, e. g. , the weight of a vehicle will separate most trucks from most cars. The statistical pattern recognition methods are very successful for those problems where suitable features can be found to differentiate between the classes. The main problem is then to discover good features and this is rarely an easy task. Most usually there is no single measurement that will do the job and the designer must try combinations of many measurements in order to find a set of features that works. It was during the "feature selection" stage of its development that pattern recognition got a reputation of being an art, rather than a science. Obvious characteristics like shape are very hard to represent numerically, so problems like recognizing triangles from circles are very hard when only statistical methods are used.

The next big advance in pattern recognition methods was when the idea of "structure" was introduced. The structure of an object can be described by telling what its parts are and how the parts are put together. If you have such a description, then you can classify an unknown object by comparing its structure with the descriptions for the different classes, e. g. , a triangle has three straight lines as its circumference and they are joined to make a closed figure. The pattern recognition problem then became to define the

appropriate parts, discover how to recognize them and discover the proper rules for combining parts to build the objects of interest. In those problems where the structural method is useful, the choice of parts is usually obvious and they are normally chosen so that they can be recognized by a statistical classifier or by template matching. Also, the rules for putting the parts together is usually clear, like straight line segments are joined end-to-end to form a triangle. However, the structural method is not the universal solution. It has its problems when observational noise is present. The effects of noise will be discussed in detail when the Bus Detector is described, but some appreciation of the noise problem can be gotten by thinking about how to recognize free-hand drawings of triangles from a structural description that uses "straight line segments."

The pattern recognition program that will be described later in this handbook uses a combination of statistical and structural methods.

## WHY USE CLASSIFIERS?

Even though pattern recognition uses the methods of statistical decision theory, it uses them in a somewhat different way than a statistician would. The key to rigorous use of statistical methods is that you know what you are measuring and you carefully define the experiment before you make any measurements! If these conditions are satisfied, then classical statistical methods are the best way to make a decision. In the problems where pattern recognition is useful, one or both of these conditions cannot be filled. The Bus Detector problem is a good example of this. A vehicle crosses a detector loop buried in the roadway. This produces a time-varying signal. The Bus Detector uses this signal to decide whether the vehicle is an urban transit bus or some other kind of vehicle. Why not use statistical decision theory? First - each time a vehicle crosses the loop we are performing an experiment over which we have almost no control! The vehicle can be moving at



any speed. It may stop over the loop and stay there for an unknown length of time. There may be two vehicles side-by-side or bumper-to-bumper, etc. All such variations introduce parameters that have to be identified ahead of time and estimated during the experiment if the statistical method is to be used. Second - knowing what we are measuring means that we have a model, parametric or statistical, that relates the experiment to the measurements. Even if we could control the vehicles, it would be impractical to develop a numerical model of how each kind of vehicle affects the loop under different weather and load conditions. And it seems obvious that a closed form solution (normal, binomial, etc.) would not work. Thus we are led to seeking a pattern recognition solution.

How does pattern recognition get around these problems? The statistical approach is to control the experiment and treat the outcome as a random variable. The pattern recognition approach is to consider the experiment itself as a random variable. It assumes that a "random" sampling of experiments will provide enough information to characterize all further experiments and that conclusions drawn from this sample will also be valid. The sample experiments are called the "training set". In the case of the Bus Detector, a loop (like the one used in the street installations) was installed in a testing area and a large number of different vehicles were driven over it. These sample experiments included busses, trucks, semi's, cars, bikes, etc. traveling at a variety of speeds. Some of them stopped over the loop, others were braking or accelerating as they passed over the loop. We tried to have the test vehicles do all of the things that vehicles on the street might do. The data base thus represents or implicitly contains a numerical model of how the vehicles affect the loop under a wide range of conditions. The pattern recognition problem is to discover a simple numerical/logical formula for this model and use it to recognize the vehicles.

## HOW IS A CLASSIFIER DESIGNED?

To answer this question we will first discuss an ideal method and show why it is impractical and then discuss a practical approximation to the ideal. We will use the bus detector problem as described in Chapter 1 as the basis for this discussion and we will assume that a sufficiently large number of experimental runs have been performed so that we have examples of all the vehicle types, speeds and paths that will be found in practice. Also assume that the waveforms from these experimental runs have been digitized at  $1/50$  second intervals (about  $1\text{-}1/2^*$  feet of vehicle motion at 60mph\*). This data base then contains or represents all the information that should be needed to classify these vehicles.

The ideal classification method is known as a "nearest neighbor" classifier. Call the digitized experimental waveforms "prototypes." Each prototype represents vehicle type  $X$  traveling at speed  $Y$  crossing the loop along path  $Z$ . The sequence of digital values  $v_1, v_2, v_3, \dots, v_n$  of a prototype waveform are a function of the conditions  $X, Y, Z$  for the experimental run that produced these values. Thus we can formally write;

$$(v_1, v_2, v_3, \dots, v_n) = f(X, Y, Z). \quad (1)$$

The basic assumption for the nearest neighbor classifier is that this function is continuous and single valued, i. e. , small changes in the experimental conditions produce small changes in the waveform and different conditions produce different waveforms. If this is true, then the function (1) can be inverted, i. e. , there is a function  $F$  such that the conditions can be determined from the waveform;

$$(X, Y, Z) = F(v_1, v_2, v_3, \dots, v_n). \quad (2)$$

\*  $1.5 \text{ ft.} = .46 \text{ m}$   
 $60 \text{ mph} = 96.6 \text{ kph}$



This function is defined as follows: Given an unknown waveform  $u_1, u_2, u_3, \dots, u_n$ , compare it with each of the prototypes and find the prototype to which it is most similar. The conditions X, Y, Z that produce the best matching prototype thus identify the unknown. [There are many ways to do the waveform comparison. Some of the most often used are: (1) sum of squares of differences (Euclidean distance),  $E = (u_1 - v_1)^2 + \dots + (u_n - v_n)^2$ , (2) sum of absolute values of differences (uniform metric or cityblock distance),  $E = |u_1 - v_1| + \dots + |u_n - v_n|$ , (3) maximum absolute difference of sample values,  $E = \max(|u_1 - v_1|, |u_2 - v_2|, \dots, |u_n - v_n|)$ . The best match is defined as the prototype that gives the smallest error E.] If the experimental data truly represents all the conditions, then this will be as good a classifier as can be found. The problem with this classifier is the large amount of computer storage and computation needed to implement it. A good estimate of the data storage needed for the bus classifier is 50 vehicles each at 10 different speeds along 5 different paths which yields 2500 prototypes. Each prototype is sampled at 1/50 sec intervals for 50 seconds (to accommodate vehicles that stop) for 2500 sample values per prototype. This requires 6.25 million computer words to store the prototypes. To compare an unknown against all of the prototypes by the second method mentioned above uses about 20 million add times. Thus a very large, very fast computer could classify one vehicle every five to ten seconds. Of course it is possible to reduce the requirements by limiting the number of conditions and the precision of the classifier. A useable set might be 20 vehicles, five paths, five speeds, 1/10 sec sample interval and 10-sec long signals. This gives 50,000 storage locations for the prototypes and 150,000 operations per decision. A good mini-computer like the IMP-16 would be able to do this in about 0.3 second per vehicle. If we allow 1/2 second spacing between vehicles (six foot spacing at 8mph) then one IMP-16 could handle one lane of traffic and it would probably take three microprocessors like the 8080 operating in parallel for each traffic lane. These reasons, storage and time, are why the nearest neighbor approach is called impractical for this problem.

Before totally dropping the nearest neighbor method we should see if there are some simple changes that might make it practical. In the example shown in Figure 3 we used normalized time and amplitude values as features. This normalization removes most of the effects of different speeds and paths and would thus reduce the number of prototypes needed. Also, if the prototypes are represented by just the peak values and times, then the amount of storage needed for each prototype is only six numbers, (two time ratios, two amplitude ratios, total time - to be used for the stopped vehicle test and the class). If 20 prototypes were enough then very little storage is needed and the computation time is reduced to less than ten milliseconds for the IMP-16. However, there is a problem with this solution. In this discussion of the nearest neighbor method we have ignored random variations and noise. During the data collection for the bus classifier design experiments were done where a single vehicle drove over the loop a number of times. The driver was told to travel at the same speed and follow the same path each time. When the normalized features for these runs are compared with each other and with runs made by other vehicles, there is as much variation among the presumably identical runs as there is between them and runs by other kinds of vehicles. This only shows that the modified nearest neighbor classifier would need more than just one prototype per vehicle class. The number of prototypes needed more like 40 to 60 to accurately distinguish between busses and all other types of vehicles. With this estimate it still looks like one IMP-16 could do the classifying for an eight lane intersection, but there would not be much time left over for other activities.

The defining characteristic of the nearest neighbor method is that it represents a class by remembering particular examples that belong to the class. This kind of representation can be very complex, as the discussion above indicates. The example shown in Figure 3 showed how straight lines could be used to simplify the class description when there are only two features. To describe or define the bus and semi regions shown in Figure 3f would require about 20 prototypes, assuming sample points are used as the

prototypes, and there would still be confusion in the overlapping region. The single straight line shown in Figure 3g separates the space into two regions in a much simpler way and it does so with only six errors. Since there is a region in the plane where we do not know whether the points are in one class or the other, any decision rule that classifies points in this region will be open to question. Without further information we cannot say whether the straight line classifier (that makes certain errors) or the nearest neighbor classifier (that retains the confusion) is closer to the real world. However, for both philosophical and practical reasons, when two descriptions are equally possible, it is best to choose the simpler one. For this reason the straight line division is preferred. The straight line can be defined by only two parameters by the equation

$$aA_1 + bT_1 = 1 \quad (3)$$

where  $A_1$  and  $T_1$  are the feature values and  $a$  and  $b$  are the parameters. This is in contrast to the 40 parameters (two coordinates for each of 20 points) that the nearest neighbor description requires. Similarly, to decide, by the nearest neighbor method, which class an unknown point with features  $A_1, T_1$  belongs to requires calculating 20 expressions of the form  $|A_1 - A_i| + |T_1 - T_i|$  and comparing their values to determine which one was the smallest. By the straight line method the class is determined by calculating the value  $x = aA + bT - 1$ . If  $x < 0$  then the point is on the bus side of the line and for  $x > 0$  the point is on the semi side of the line. ( $x = 0$  for points on the line.)

The expression  $aA_1 + bT_1 - 1$  is called a "Linear Discriminant Function." It discriminates between points in the two classes by being positive in the region belonging to one of the classes and negative in the region of the other class. If we used all four of the normalized features then a linear discriminant function of these four variables will have the form

$$pA_1 + qA_2 + rT_1 + sT_2 - 1 = D(A_1, A_2, T_1, T_2) . \quad (4)$$



The equation

$$D(A_1, A_2, T_1, T_2) = 0 \quad (5)$$

is the equation for a "hyperplane" in the four dimensional space. One of the complications that comes with using more than two features is that the points cannot be plotted on graph paper. (K&E has graph paper that can be used to plot points in three dimensions but four dimensional graph paper has not been tried yet.) Even though we cannot plot points in these higher dimensional spaces, we can and do still use geometric concepts and terms to discuss and solve pattern recognition problems. Thus in three dimensions a linear discriminant defines a plane, in four dimensions it defines a hyperplane. We talk about the two sides of a line or two sides of a plane - in each case the negative side contains the origin [ $x(0,0) = a \cdot 0 + b \cdot 0 - 1 = -1$  in two dimensions] and the other side is the side away from the origin. Similarly, in four dimension we can talk about the two sides of a hyperplane. A hyperplane divides the four dimensional space into two parts, one of which contains the origin and the other part does not.

To separate two classes in four dimensional space we have to find a hyperplane that divides the two classes like the line does in Figure 3g. The formulas for doing this are in the CTC program and the method is explained in detail in Chapter 3. The ideas behind the mathematical method are quite simple. To discuss them we need another analog to a line in two dimensions. The notation using the normalized bus detector features  $A_1, \dots, T_2$  is a little bit clumsy for this discussion so we will use the more familiar symbols  $(x, y)$  for the coordinates of a point in two dimensional space and  $(x, y, z, w)$  for four dimensions. A line in two dimensions is then given by the equation

$$ax + by - 1 = 0 \quad (6)$$



and a hyperplane is given by

$$ax + by + cz + dw - 1 = 0 \quad (7)$$

The corresponding linear discriminant functions are

$$D_2(x, y) = ax + by - 1 \quad \text{and} \quad D_4(x, y, z, w) = ax + by + cz + dw - 1 \quad (8)$$

where the subscripts indicate the dimension of the space. As before, the equations for the line and the hyperplane are

$$D_2(x, y) = 0 \quad \text{and} \quad D_4(x, y, z, w) = 0 \quad (9)$$

respectively. These two functions are really normalized distance functions whose unit of measurement is the distance from the line (or hyperplane) to the origin. To justify this interpretation, note that the distance from the line to any point on the line is zero, and for any point  $(x, y)$  on the line  $D_2(x, y) = 0$ . Similarly for the hyperplane. Also, the value of the discriminant function at the origin  $(0, 0)$  is  $D_2(0, 0) = -1$  (one unit in the negative direction) and similarly for the hyperplane. Thus, if we have a discriminant function, whether it be in two dimensions or in four dimensions, the values of the function are simply distances relative to the location of the line or hyperplane. Distances can be plotted very easily on graph paper or they can be listed in a table. Either way, by using the discriminant function values for the sample points we can "see" exactly how the sample points are located in the space, whether it is a two dimensional space or a four dimensional space. (In the CTC program these are called "tradeoff" listings.)

We will next consider how these distance plots will help to design a classifier. The discussion applies to any number of dimensions (CTC can handle up to ten dimensions) so we will drop the terms line and hyperplane and use the term "boundary" for the surface defined by the discriminant function. Suppose we

are given a set of points representing samples from two different classes. Call these points  $P_1, P_2, P_3, \dots, P_n$ . We will name the classes "1" and "2" rather than bus, semi, etc. Next, define a new function, called the class function  $C(P)$ , by

$$C(P) = 1 \text{ if } P \text{ is in class 1, } C(P) = 2 \text{ if } P \text{ is in class 2} \quad (10)$$

and let  $C_i = C(P_i)$ . If we could find a formula for the class function then the classifier design would be done, because to classify any point  $P$  we would only have to calculate the value  $C(P)$  of the class function and that would be the class name. The problem with trying to do this is that the class function only defined at the sample points  $P_i$  and we would like to be able to classify other points that we have not seen before. One way to do this is to treat the problem of classifying the unknown points as a problem of interpolating between the known points to find the values at the unknown points. In particular, linear interpolation would be very nice because linear functions are so easy to compute. So, for the purpose of discussion, let us assume that we can find a linear interpolating function  $L(P)$  that either fits the known sample points  $P_i$  exactly [i. e.,  $L(P_i) = C_i$ ] or, if an exact fit is not possible, then one that fits the known sample points approximately [i. e.,  $L(P_i) \approx C_i$ ]. Since the interpolating function is continuous, there will be points where it has values other than 1 and 2. For example, suppose  $L(P_i) = 1$  and  $L(P_j) = 2$ , then, for the point  $P = P_i + P_j / 2$  that lies halfway between  $P_i$  and  $P_j$ , we will find  $L(P) = 1.5$  because of the linear properties. Now, suppose that by some act of providence all of the known points in class 1 are close to each other and all the known points in class 2 are close to each other and that class 1 and class 2 are far apart. (If this is true then we say that the known sample points are "clustered" and the classes are "widely separated.") Since the known points in class 1 are clustered, we would expect that any new, unknown point belonging to class 1 will either lie inside the cluster or close to it. Therefore the interpolating function will have values close to 1 for all points in class 1. Similarly the interpolating function will have values close to 2 for points in class 2. Thus we could define a classification rule

$$\begin{aligned} &\text{"assign } P \text{ to class 1 if } L(P) < 1.5 \text{ and} \\ &\text{assign } P \text{ to class 2 if } L(P) > 1.5" \end{aligned} \quad (11)$$

There is a very simple relationship between this interpolating function  $L(P)$  and the linear discriminant function  $D(P)$  discussed earlier;

$$D(P) = -\{L(P) - 1.5\} / \{L(0) - 1.5\} \quad (12)$$

where 0 denotes the origin. We thus see that the boundary  $D(P) = 0$  determined by this method lies halfway between the two clusters, since  $L(P) = 1.5$  happens for points that are halfway between.

Of course we cannot expect that the points obtained by real world experiments will always form nice clusters. Instead we expect experimental data to be scattered and have overlapping class regions as in Figure 3. But the interpolating function idea still works in these situations. Regardless of how badly scattered or overlapped the experimental classes are, the interpolating function will still be approximately 1 for class 1 and approximately 2 for points in class 2 and the boundary  $L(P) = 1.5$  will still split the space into two parts. The difference in this case is that the interpolating function does not match the class function as well and there are experimental points that lie on the wrong side of the boundary and are thus mis-classified. But we do not really care how closely the interpolating function fits the class function - the important thing is how well the points are classified, i. e., how well the boundary separates the classes! As we discussed earlier, when the experimental points are in overlapping regions then the classifier will probably make some mistakes just because of the way the classes are defined by the experimental data. In such cases, a rational course of action is to attempt to minimize the errors without making the solution any more complex than can be justified by problem and the data.

The preceding discussion assumed that a linear interpolating function could be found. Next we will look at how to find such a function. For this discussion we will go back to the notation in Equation (8). Let the sample points  $P_i$ ,  $i=1, \dots, n$  have the coordinates  $P_i=(x_i, y_i, z_i, w_i)$  and let  $a, b, c, d, e$  be the coefficients of the interpolating function;

$$L(P_i) = ax_i + by_i + cz_i + dw_i + e. \quad (13)$$

The class function values  $C_i = 1$  or  $2$  are the ideal values for  $L(P_i)$ . We can write this ideal condition as

$$\begin{aligned} ax_1 + by_1 + cz_1 + dw_1 + e &= C_1 \\ ax_2 + by_2 + cz_2 + dw_2 + e &= C_2 \\ . & \\ . & \\ . & \\ ax_n + by_n + cz_n + dw_n + e &= C_n \end{aligned} \quad (14)$$

If there are only five sample points then we will have a system of five equations in five unknowns  $a, b, c, d, e$  and, except for certain singular cases, the equations will have a solution. In general we want and will have many more than this minimum number. In that case the coefficients are found by the least squares method. Define

$$E = (ax_1 + by_1 + cz_1 + dw_1 + e - C_1)^2 + \dots + (ax_n + by_n + cz_n + dw_n + e - C_n)^2. \quad (15)$$

Each of the squared terms is the difference or error between the value of the interpolating function and the desired value. The best solution for the coefficients  $a, \dots, e$  would make each of these errors equal to zero and if



that is not possible then the next best is to make the sum of squared errors (15) as small as possible. We can find the values of  $a, \dots, e$  that minimize  $E(a, \dots, e)$  by solving the set of linear equations shown below:

$$\begin{aligned}\frac{\partial E}{\partial a} &= 0 \\ \frac{\partial E}{\partial b} &= 0 \\ \frac{\partial E}{\partial c} &= 0 \\ \frac{\partial E}{\partial d} &= 0 \\ \frac{\partial E}{\partial e} &= 0\end{aligned}\tag{16}$$

It is not necessary, for the present discussion, to deal with the way these can be solved. The CTC program does the solution in what is called the method 1 classifier, so we will let the matter rest.

Once the coefficients are known, the decision rule can be implemented by Equation (11) and the classifier design is finished. This procedure yields a two class classifier that is close but not perfect. There are a number of simple ways to use this classifier as an initial approximation and, by modifying the coefficients, obtain better performance. These methods are discussed in Chapter 3.

## MULTICLASS CLASSIFIERS

The final topic for this chapter is "multi-class classifiers." So far we have only talked about separating or distinguishing between two classes of objects, like bus versus semi or bus versus not-bus. The bus detector classification

is stated as a two class problem - "Recognize busses and reject everything else" - so it is natural to think that it can be solved as a two class problem using the method given above. But this is not the case. The "class" of vehicles denoted by the term "not-bus" is built up from the classes known as "truck," "semi," "moving van," "school bus," "VW," "Torino," "motor cycle," "Step Van" and on and on. Their only points of similarity are that they are all motorized vehicles and they are all not-busses. If we had a sensor that could measure a feature called "business" then we would have a reason to think that there is a natural reason for defining a natural not-bus class. But the sensor we are using does not measure "business," it measures a physical quantity related to the structure of the vehicles it is sensing. The natural way to define classes relative to this sensor is to group vehicles that have similar measurements into a class and require that if two vehicles have different measurements they must be placed into different classes. (This is the reason that "clusters" of points were talked about earlier.) Therefore if we put vehicles with similar structures together in classes we will have a multi-class problem involving "bus," "truck," "semi," etc. rather than a two class problem.

In designing the bus detector classifier we found that busses from different manufacturers formed separate clusters and that it was even possible to distinguish between different model years of the same make. In working a classifier problem it is generally a good idea to keep all of the distinctions by defining different classes for each distinguishable kind of object. Thus in the bus detector data base the vehicles are labeled as "bus 1," "bus 2," "bus 3," "truck," "semi," etc. Furthermore the moving and stopped vehicles were labeled separately so that we had "bus 1 - moving," "bus 1 - stopped," etc. as the predefined classes. With these a priori distinctions included in the data it is then easy to combine small classes into larger ones if the analysis shows that they are similar in terms of the feature values. The opposite, finding separate clusters in data that is labeled the same, is a very difficult problem and should be avoided when possible.

Thus we see that the multi-class problem is the rule and the two class problem is the exception. Fortunately a multi-class problem can be solved as a sequence of two class problems, as shown by the following example: suppose we have three classes of objects, call the classes A, B, C. Further suppose that we have three classifiers  $K_{ab}$ ,  $K_{ac}$ ,  $K_{bc}$  each of which is a two class classifier that can distinguish between objects in the classes indicated by the subscripts. Then for an object that is in either A or B classifier  $K_{ab}$  will make the correct decision, but for an object in C classifier  $K_{ab}$  may be right or it may be wrong - we write this as  $K_{ab}(A) = A$ ,  $K_{ab}(B) = B$ ,  $K_{ab}(C) = ?$ . Now suppose that we have an object  $x$  whose class is unknown. We can use each of the classifiers to classify the  $x$ , even though we do not know which ones will give the right answer. What we do know is that two of the classifiers will be right and the third one may be right and it may be wrong. Thus we can find the right answer simply by taking a majority vote of the three classifiers.

The majority vote method of combining two class decisions to make a multi-class decision is a sufficient solution to the problem, that is, it will do the job but we may be able to get by with a simpler method. The bus detector shows why it may be desirable to look for an easier or simpler way. For the bus detector design we considered 20 vehicle classes at the start. This included six bus classes (three different busses, moving and stopped) and fourteen others (seven vehicle types, moving and stopped). By the majority vote method we would have to classify all possible pairs of classes and this would require  $(20)(19)/2 = 190$  two class decisions! There must be a better way. And there is. For the bus detector we start by first deciding whether the vehicle was moving or had stopped. These two cases are then treated by separate classifiers. Potentially, each of these classifiers would have to consider 45 two class decisions (possible pairs of ten vehicle types). While this would be a saving over the 190 pairs discussed above, it was possible to simplify the solution even more. For moving vehicles it was found that the semi's and moving vans could be distinguished, as a class,

from all the other vehicles. (This might be expected because of the number of axles.) It was also found that the semi's, moving vans and busses could be grouped together into a class and this class could be distinguished from the remaining vehicles. (Again this might be expected on the basis of vehicle size.) We were thus able to construct two classifiers:

$K_1$ : Two axle versus three axle

$K_2$ : Large vehicle versus small vehicle

The final classifier then combines these two decisions and defines a bus as a "large, two-axle vehicle." A similar separation was possible for the stopped vehicles. (Actually the pairwise separations indicated above were not perfect, but the classifiers, when combined, identified the busses correctly.)

With this introduction to the problems and methods of classifier design finished, we will next consider a simple numerical example to show the methods in action. The problem of designing the bus detector/classifier is taken up again in Chapter 6.



## CHAPTER 3

### EXAMPLES OF CLASSIFYING MULTIDIMENSIONAL DATA

#### 3.1 INTRODUCTION

To consider interactive approaches to classifier design we must first discuss some properties of sample data distributions and how they are used in classifier design. We also need some ideas involving linear (or piecewise linear) discriminants and how they are used to partition the feature space.

In this chapter, the necessary properties of sample data distributions are discussed and numerical examples of the linear, piecewise linear, and layered machine classifiers given. This will provide the background for using interactive techniques and for using the Cascaded Threshold Classifier (CTC) program.

#### 3.2 SAMPLE DATA DISTRIBUTIONS

For most problems appropriate for classifier solutions, the probability distribution from which the experimental data are drawn is unknown. Only the class membership for each data point is known. Thus, one must make assumptions concerning the underlying probability distribution using properties of the sample data distribution to justify these assumptions. The assumptions are then used to guide the classifier design process.

The most common assumption is that the probability density for each pattern class is Gaussian. The distribution function for class  $C_i$  is then given by:

$$P(X|C_i) = \left[ (2\pi)^n \det V_{X,i} \right]^{-1/2} \exp \left[ -\frac{1}{2} (X-m_i)^T V_{X,i}^{-1} (X-m_i) \right] \quad (17)$$

where  $m_i$  is the mean and  $V_{X,i}$  is the covariance of  $C_i$ . These are estimated from the experimental data by the formulas:

$$\hat{m}_i = \frac{1}{M_i} \sum_{j=1}^{M_i} X_j^{(i)} \quad (18a)$$

$$\hat{V}_{X,i} = \frac{1}{(M_i - 1)} \sum_{j=1}^{M_i} (X_j^{(i)} - \hat{m}_i) (X_j^{(i)} - \hat{m}_i)^T \quad (18b)$$

The linear discriminant (see Subsection 3.3.1 and Appendix A) can be shown to be the optimal means of deciding between two classes  $C_1$  and  $C_2$  each having the density given in Equation (17), but with unequal mean vectors. In two dimensions, the linear discriminant is a line and, in three dimensions, it is a plane. In any case, the linear discriminant separates the feature space into two parts. One part contains the points most likely belonging to class  $C_1$ , the other to  $C_2$ . To classify a sample, the classifier determines on which side of the discriminant the data point lies.

A common approach to classifier design is to assume that all classes are described by unimodal Gaussian densities as in Equation (17), and then use the data to compute coefficients for the linear discriminants between each pair of classes. The unimodal assumption can be weakened to include multi-modal classes described by the density (mixture density)

$$P(X|C_i) = \sum_{\ell=1}^{K_{C_i}} P_{i,\ell} P(X|C_{i,\ell}), \quad P_i = \sum_{\ell} P_{i,\ell} < 1, \quad (19)$$

where each  $P(X|C_{i,\ell})$  can be expressed as in Equation (17). The solution, for classification, is to split each such class (multi-modal class) into the appropriate  $K_{C_i}$  sub-classes, extending the original  $N$  class problem to an  $\hat{N} = \sum_{i=1}^N K_{C_i}$  class problem, and use linear discriminants. The original  $N$  class problem is then solved by piecewise linear discriminants. This use of subclasses is further explained in this subsection, and Subsections 3.3 and 3.5.

The validity of the unimodal gaussian assumption or the multimodal gaussian assumption rests upon the following two properties:

1. Complexity - the complexity of a sample data distribution comes from two sources: a) inter-class complexity; and b) intra-class complexity. Loosely speaking, inter-class complexity is related to how classes overlap and the shape of the overlap regions. Intra-class complexity is related to the shape and modality of a class. Intra- and inter-class complexity are not independent of one-another in their effects on classifier design.
2. Dimensionality - Experimental data are used to design classifiers, but the raw data are usually transformed before they are used for classification. While the raw data values (measurements) may be mutually independent, the transformed values (features) may be mutually dependent. If the features are dependent, then they lie in a lower dimension sub-space and the distribution is not gaussian. Intermediate between independent and dependent features is the general case of correlated features, where the correlation lies between zero and unity in magnitude. If the correlation is near unity, or if the lack of correlation is due only to noise, then it is often advisable to remove one or more of the correlated features so as to make the data distribution more like a gaussian distribution.

To aid in discussing complexity, consider the sample distribution shown in Figure 5a.

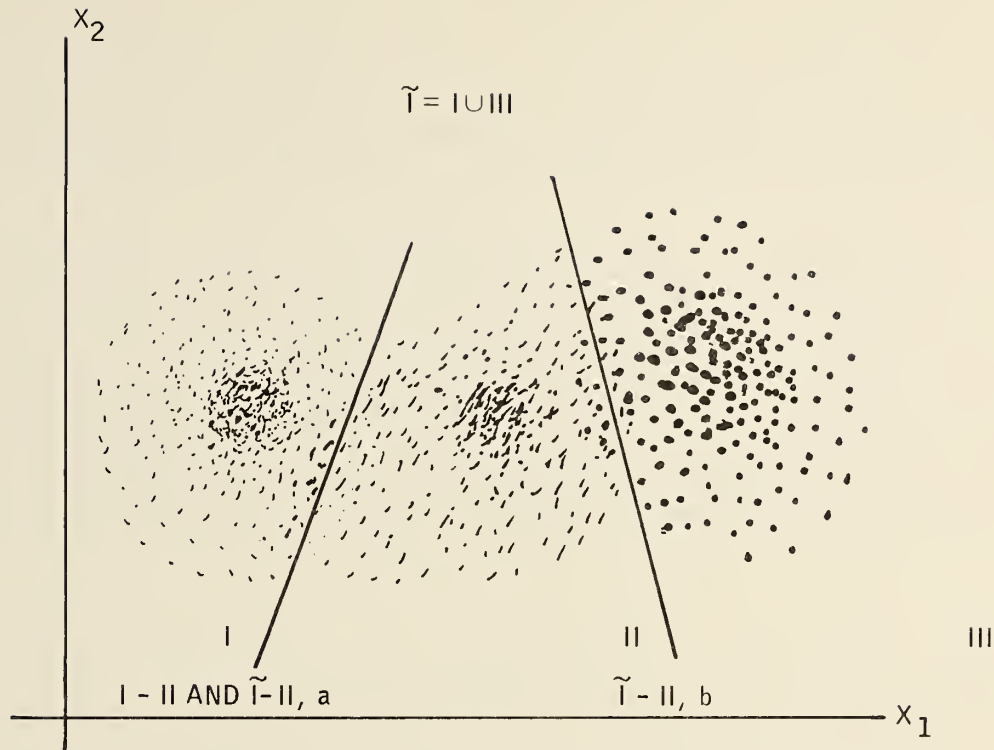


Figure 5a. Example of Low Complexity Distribution

Figure 5a, classes I, II and III, are examples of distributions with low complexity relative to a linear classifier. That is, they are approximately gaussian. The 'ball-like' cluster of data points for each class conforms to the distribution one would expect for a class with a unimodal gaussian density; thus low intra-class complexity.

Further, there is little overlap between the classes, hence there is a low inter-class complexity. Figure 5a, classes  $\tilde{I}$  and II, illustrates the analogous low-complexity case relative to a piecewise linear classifier since class  $\tilde{I}$  is multi-modal gaussian.

Figure 5b shows the effect of increasing inter-class complexity (as compared to Figure 5a, classes I and II) while maintaining intra-class simplicity. A linear classifier cannot separate classes I and II. Note that unequal class means does not guarantee inter-class simplicity. Any instance for which the distance between means is on the order of the average point spread about



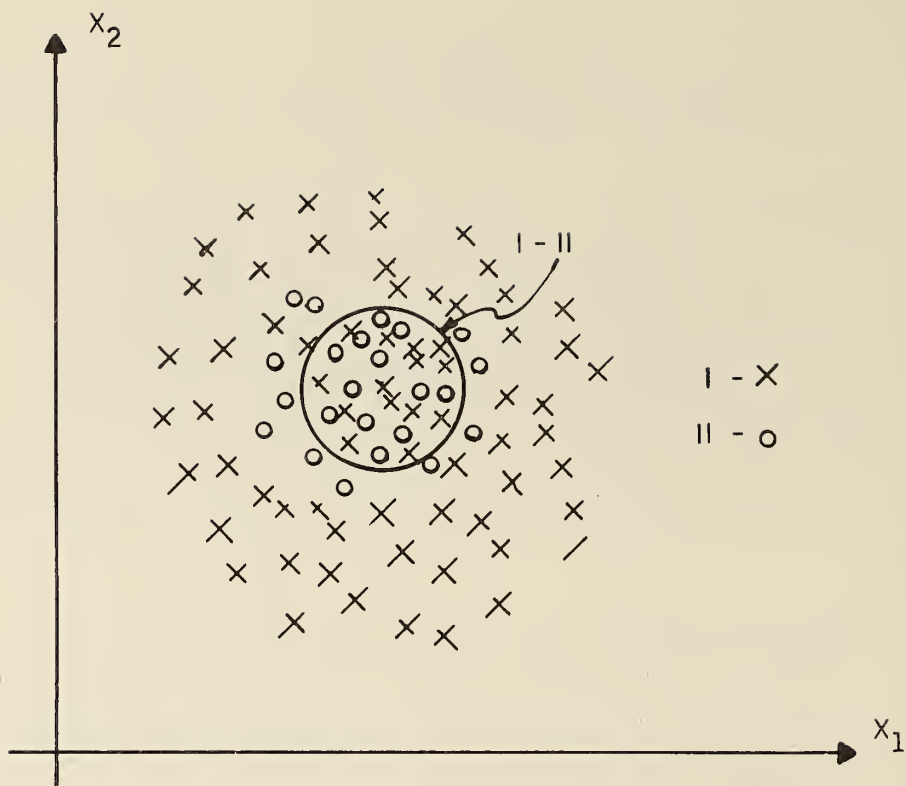


Figure 5b. Low Intra-class Complexity with High Inter-class Complexity

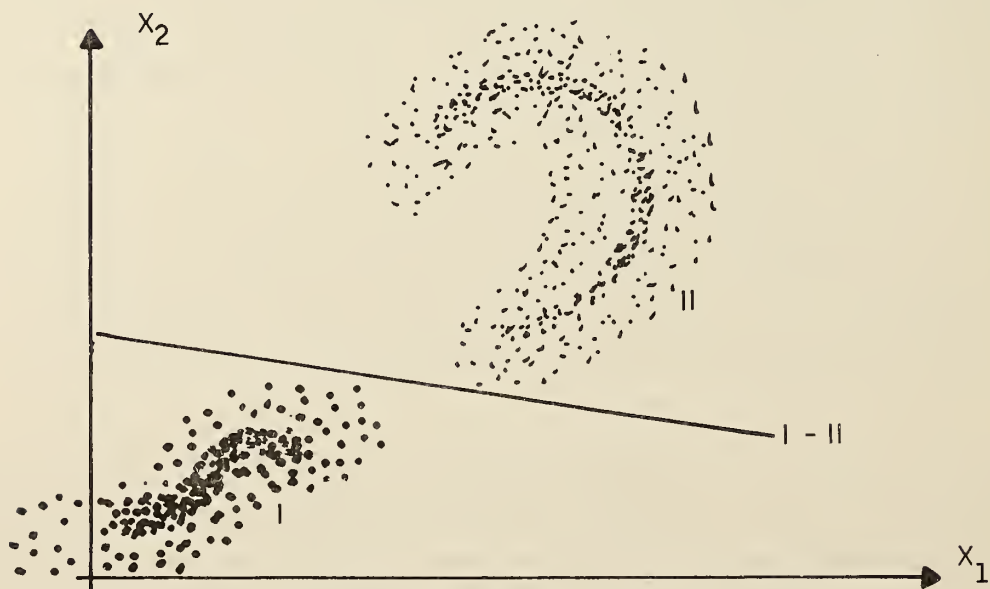


Figure 5c. High Intra-class Complexity with Low Inter-class Complexity

the mean for the classes involved, the linear classifier's performance will be poor (see Appendix A.)

Figure 5c shows the effect of increasing intra-class complexity while maintaining inter-class simplicity. The distribution of each of the classes I and II in Figure 5c does not conform to a unimodal gaussian density. However, the classes are well separated and one can use a linear classifier in spite of the gaussian assumption failing. (Note that the resulting classifier is not optimum for the distribution shown, but may achieve a low enough error rate nonetheless.)

Finally, Figure 5d illustrates a case of high overall complexity, both inter- and intra-class, relative to the gaussian assumption. Relaxing the unimodal assumption to a multimodal assumption might allow a piecewise linear classifier to be designed for the example data, but the choice of sub-classes is not clear. One could separate classes I and II of Figure 5c into subclasses by trial and error, but this is obviously stretching the validity of the gaussian assumptions. A case such as Figure 5c generally requires a more powerful approach to classification, beyond the scope of this handbook, or a new choice of features if one insists upon linear or piecewise linear classifiers.

Let us return to Figure 5a, classes  $\tilde{I}$  and II. For this case, using the unimodal assumption, the overall complexity is high. Classes  $\tilde{I}$  and II have nearly equal means, and class  $\tilde{I}$  does not conform to a distribution resulting from a unimodal gaussian density. The proximity of means for classes  $\tilde{I}$  and II relative to the average spread about the mean implies high inter-class complexity. Thus, the actual intra-class complexity of  $\tilde{I}$  shows up as inter-class complexity unless it is recognized that  $\tilde{I}$  is multimodal. Under the multimodal assumption, the complexity of the problem is low. This illustrates the need for checking intra-class as well as inter-class complexity.

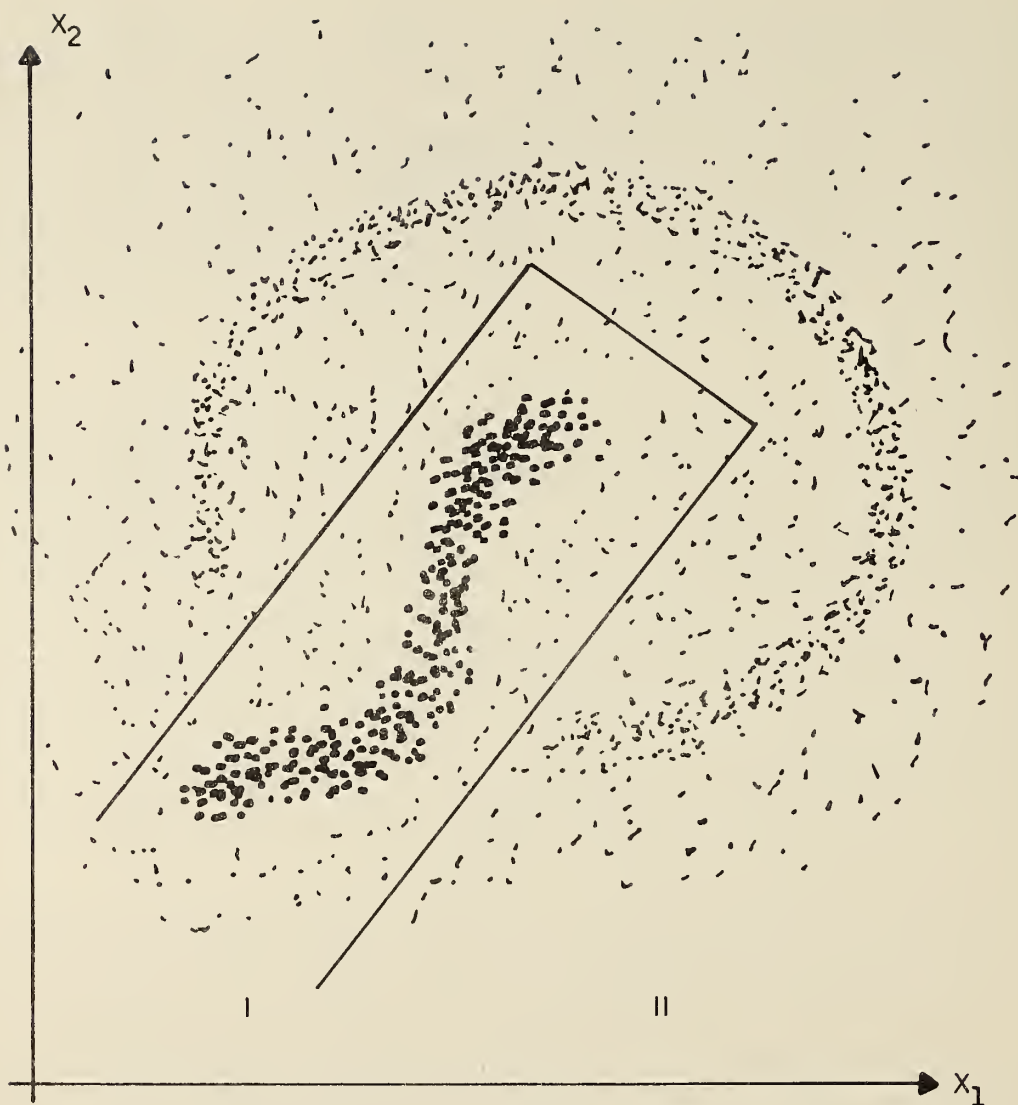


Figure 5d. High Inter- and Intra-class Complexity

The complexity of the sample data distributions in the previous examples was gauged by 'eyeball' methods, using the fact that for  $n=2$ , all of the details of the distribution can be shown in a scatter plot (Figures 5a through 5d). Obviously, the generation of scatter plots, while easy for  $n=1$ , and 2, is difficult for  $n=3$  and impossible for  $n$  greater than 3. Generally, one must rely upon projections onto a two or three dimensional subspace for cases when the feature space of a dimension greater than or equal to 3. Thus, methods other than scatter plotting must generally be used to gauge complexity.

From the example in Figures 5a through 5d, we can see that one approach to measuring complexity is to

1. Consider a specific form of classifier (a certain set of assumptions concerning the distribution)
2. Measure complexity with respect to that classifier (with respect to those assumptions).

More general techniques for gauging complexity independently of a specific classifier have been proposed, but such techniques are beyond the scope of this handbook (see reference by Meisel).

Under the approach to complexity described in this handbook, the simplest distribution is the one matching the conditions on class probability densities imposed by the specified classifier. (This makes the classifier optimal in the minimum probability of error sense). Conversely, the most complex distribution is one for which the conditions do not match, and the assumptions on the distribution for the given classifier are completely erroneous.

We are specifically interested in verifying the unimodal or multimodal assumptions. Note that our interest is not in the strict verification of these assumptions, only that the distributions have enough of the properties of gaussian distributed data for a linear or piecewise linear classifier to be feasible.



Since we must work with data for which  $n$  is greater than three, numerical measures of complexity must be used to augment projection scatter plotting. Numerical measures of complexity are based on the mathematical form of the classifier considered (see Appendix A). Complexity is gauged by the CTC program using the following computations:

1. Euclidean distance between class (and subclass) means, and angles between class means referred to the overall mean
2. Statistical scatter computations; coveriance, correlation coefficients, standard deviations for the data
3. Projections onto subspaces
4. Property listing (a single-dimension scatter plotting technique, quite powerful when used in conjunction with computation 3)

The computations are described in detail in Appendix C. Property listing is described in Chapter 4, with the description of '-7' control cards. These methods for gauging complexity are well suited to verifying properties of the data relative to the gaussian assumptions, and deciding how to assign subclasses in the case of multimodal classes.

From Appendix B, the multimodal assumptions result in an extended problem for which the unimodal assumptions are used. From Appendix A, Equation (A-10),

$$r_{jk} = \left( \sum_{i=1}^n \Delta m_i^2 / \tilde{\sigma}_i^2 \right)^{-1} \quad (20)$$

where

$$\Delta m_i = \hat{m}_{j,i} - \hat{m}_{k,i}$$

$$\tilde{\sigma}_i^2 = (M_j \hat{\sigma}_{j,i}^2 + M_k \hat{\sigma}_{k,i}^2) / (M_j + M_k)$$

is a measure of inter-class complexity that can be easily computed from the results of computations 1 and 2. Intra-class shape complexity can be measured by the standard deviations and correlation coefficients for the class, while modality (another aspect of intra-class complexity) can be estimated by using the property listing computation together with the projection computation. This is described more completely in the Subsection 3.3 example.

Dimensionality of the data, in general, depends on the number of samples per class and the number of classes as well as the number of features used. Requirements on dimensionality necessary to reliably gauge complexity and classify the data can be stated as inequalities involving these three quantities.

In most applications, one has a limited number of datapoints and, hence, is required to extract as much information as possible from them, even though they may be sparse in the decision space. If it can be determined that the samples are clustered in small regions of the decision space, or have high localized densities in the space, the curse of dimensionality can be tempered by considering these local regions in designing the classifier.

As a rule of thumb, in higher dimensional spaces, the possibility for complex sample distributions increases, and algorithms based on a mental image of unimodal (or simple) clusters of sample points should be used with caution or not at all (this is the reason for gauging complexity).

A rough indicator of the sample density in the decision space is the  $M_i/n$  ratio, where  $M_i$  is the number of data points in class  $C_i$  (or subclass  $C_i^j$ ) and  $n$  is the dimension of the space. An example of blindly using a set of samples with a small  $M_i/n$  ratio is given in Meisel where it is shown experimentally that  $M_i/n$  should be in the range 3 to 5 at least (the larger the ratio the better) to

prevent an apparently significant classification of samples from a single class into more than one class. Another experiment has shown that for fixed  $M_1$  (hence fixed  $M$ ), as  $n$  is increased the performance of a pattern recognizer increases to a maximum number of correctly classified samples, and then decreases. Thus,  $n$  must reflect the 'intrinsic dimensionality' of the data for the best classifier performance.

Although intrinsic dimensionality is difficult to determine (see Fukunaga, Hughes, etc.), a closely related constraint that is easier to use is the  $N/n$  ratio. ( $N$  is the number of classes.) In general, one must have  $n$  greater than or equal to  $N-1$  at the beginning stages of classifier design. If the resulting  $n$  is larger than the intrinsic dimensionality of the classification problem, certain characteristics of the data distribution will reflect this fact. However, if  $n$  is too small, one must make new measurements to get more features. In addition,  $n$  smaller than the intrinsic dimensionality of the problem is manifested by high complexity of the sample distributions. The above constraint can most easily be understood by considering the ways in which the means of  $N$  classes can be distributed in  $X_f$ . The most general arrangement of  $N$  points requires a set of vectors spanning  $N-1$  space to retain the spatial relations between the points. For example, two points define a line (if they are equal, the degenerate case of a line, a point), three points define a plane (or in the degenerate case, a line or a point), and so on. (Note that, in each of these cases,  $N-1$  dimensions were required to describe the most general arrangement of  $N$  points.)

### 3.3 A THREE-CLASS NUMERICAL EXAMPLE

The example in this section illustrates a number of different analytical methods for solving classifier design problems. The methods are based on quantities that are calculated by the Cascaded Threshold Classifier program. Mathematical derivations for these quantities are given in

Appendices A, B and C, and Chapter 4 discusses how to use the program to obtain them.

Three different ways to solve this problem are discussed.

- Multiple linear discriminant solution
- Piecewise linear solution using subclasses
- A layered machine solution

In each case the complexity and dimensionality of the data are tested by the methods discussed earlier.

### 3.3.1 The Multiple Linear Discriminant Solution

Table 1 lists the data that will be used for this example, which are then plotted in Figure 6. In using the CTC program, the following computational steps are normally performed:

- STEP 0 - Compute apriori probabilities (class weights)
- STEP 1 - Distance computations
- STEP 2 - Statistical scatter computations
- STEP 3 - Data listings

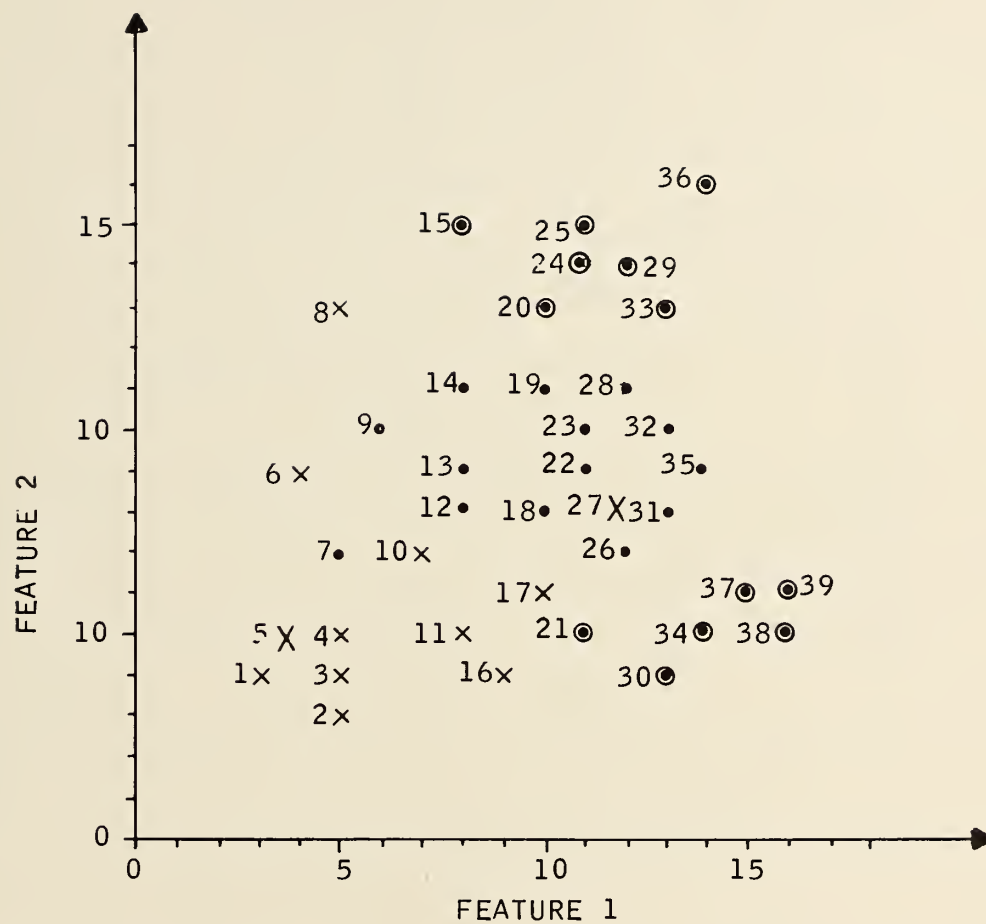
The gaussian assumption is tested by using the results of steps 1, 2 and 3 (see Appendix C for details) as follows (For this example feature 1 =  $x$  and feature 2 =  $y$ ):



TABLE 1. TEST CASE 1 DATA, NUMERICAL (AS GIVEN BEFORE DESIGN PROCESS)

Item	Feature 1	Feature 2	Class
1	3	4	1
2	5	3	1
3	5	4	1
4	5	5	1
5	4	5	1
6	4	9	1
7	5	7	2
8	5	13	1
9	6	10	2
10	7	7	1
11	8	5	1
12	8	8	2
13	8	9	2
14	8	11	2
15	8	15	3
16	9	4	1
17	10	6	1
18	10	8	2
19	10	11	2
20	10	13	3
21	11	5	3
22	11	9	2
23	11	10	2
24	11	14	3
25	11	15	3
26	12	7	2
27	12	8	1
28	12	11	2
29	12	14	3
30	13	4	3
31	13	8	2
32	13	10	2
33	13	13	3
34	14	5	3
35	14	9	2
36	14	16	3
37	15	6	3
38	16	5	3
39	16	6	3

$$M_1 = 12, M_2 = 14, M_3 = 13$$



KEY		
CLASS 1	x	TYPE 1
CLASS 2	•	TYPE 2
CLASS 3	⊙	TYPE 3 AND TYPE 4

Figure 6. Test Case 1 Data Scatter Plot

STEP 0. A priori probabilities (class weights):

$$P(C_i) = \frac{M_i}{M} \Rightarrow P(C_1) = P_1 = \frac{12}{39} = 0.31$$

$$P(C_2) = P_2 = \frac{14}{39} = 0.36$$

$$P(C_3) = P_3 = \frac{13}{39} = 0.33$$

STEP 1. Distance computations

$$\hat{m}_1 = (\bar{x}, \bar{y}) = (6.417, 6.083)$$

$$\hat{m}_2 = (10.07, 9.143)$$

$$\hat{m}_3 = (12.62, 10.08)$$

$$m_{TOT} = (9.795, 8.513)$$

$$d(\hat{m}_k, \hat{m}_j) = \begin{cases} 4.766; i = 1, j = 2 \\ 7.374; i = 1, j = 3 \\ 2.710; i = 2, j = 3 \end{cases}$$

$$\theta_{ij} = \begin{cases} 149.4^\circ; i = 1, j = 2 \\ 173.3^\circ; i = 1, j = 3 \\ 37.29^\circ; i = 2, j = 3 \end{cases}$$

STEP 2. Statistical scatter computations

$$\hat{V}_1 = \begin{bmatrix} 7.076 & 0.4653 \\ 0.4653 & 7.243 \end{bmatrix} \quad \rho_{12}(1) = 0.064999$$

$$\hat{V}_2 = \begin{bmatrix} 6.923 & 0.3469 \\ 0.3469 & 1.837 \end{bmatrix} \quad \rho_{12}(2) = 0.09729$$

$$\hat{V}_3 = \begin{bmatrix} 5.314 & -6.124 \\ -6.124 & 21.46 \end{bmatrix} \quad \rho_{12}(2) = -0.5736$$

STEP 3. Property listing

- on Feature 1 see Table 2
- on Feature 2 see Table 3.

Note that the data in Table 1, have  $n = 2$ ,  $n = 3$ ,  $M_1 = 12$ ,  $M_2 = 14$ ,  $M_3 = 13$ , hence,

$$\frac{N}{n} = N-1 \text{ (as integer values)}$$

$$\frac{M_i}{n} \geq \frac{M_1}{2} = 6$$

and the dimensionality requirements are satisfied.

Thus, the above computations are based on a statistically significant sample, and the number of features allows the most general distribution of three classes (co-planar). We can conclude that a reliable decision rule can possibly be developed using the data.

To gauge the inter-class complexity, the results of steps 1 and 2 are used together with equation (20) to get

$$r_{jk} = \begin{cases} 0.25; j = 1, k = 2 \\ 0.14; j = 1, k = 3 \\ 0.88; j = 2, k = 3 \end{cases} \quad (21)$$

as follows:

$$\begin{aligned} \bullet \quad j = 1, k = 1 & - \Delta m_1^2 = (\hat{m}_{1,1} - \hat{m}_{2,1})^2 = (6.417 - 10.07)^2 = 13.34 \\ & \Delta m_2^2 = (\hat{m}_{1,2} - \hat{m}_{2,2})^2 = (6.083 - 9.143)^2 = 9.36 \\ \tilde{\sigma}_1^2 & = (12\hat{\sigma}_{1,1}^2 + 14\hat{\sigma}_{2,1}^2)/26 = \frac{(12 \cdot 7.076 + 14 \cdot 6.923)}{26} \\ & = 6.99 \\ \tilde{\sigma}_2^2 & = (12\hat{\sigma}_{1,2}^2 + 14\hat{\sigma}_{2,1}^2)/26 = \frac{(12 \cdot 7.243 + 14 \cdot 1.837)}{26} \\ & = 4.33 \end{aligned}$$



TABLE 2. TRADEOFF FOR FEATURE NUMBER 1

DATE	610	ID	TC1AM1	NUMBER	1	CL	SCORE	0	1	2	3
TRADEOFF	NO	ID	TY								
1	1	1	4	1	0	0	1	3.0000	1		
2	6	6	4	6	0	0	1	4.0000	2		
3	5	5	4	5	0	0	1	4.0000	3		
4	8	8	4	8	0	0	1	5.0000	4		
5	7	7	5	7	0	0	2	5.0000		1	
6	4	4	4	4	0	0	1	5.0000	5		
7	3	3	4	3	0	0	1	5.0000	6		
8	2	2	4	2	0	0	1	5.0000	7		
9	9	9	5	9	0	0	2	6.0000		2	
10	10	10	4	10	0	0	1	7.0000	8		
11	15	15	7	15	0	0	3	8.0000			1
12	14	14	5	14	0	0	2	8.0000		3	
13	13	13	5	13	0	0	2	8.0000		4	
14	12	12	5	12	0	0	2	8.0000		5	
15	11	11	4	11	0	0	1	8.0000	9		
16	16	16	4	16	0	0	1	9.0000	10		
17	20	20	7	20	0	0	3	10.0000			2
18	19	19	5	19	0	0	2	10.0000		6	
19	18	18	5	18	0	0	2	10.0000		7	
20	17	17	4	17	0	0	1	10.0000	11		
21	25	25	7	25	0	0	3	11.0000			3
22	24	24	7	24	0	0	3	11.0000			4
23	23	23	5	23	0	0	2	11.0000		8	
24	22	22	5	22	0	0	2	11.0000		9	
25	21	21	6	21	0	0	3	11.0000			5
26	29	29	7	29	0	0	3	12.0000			6
27	28	28	5	28	0	0	2	12.0000		10	
28	27	27	4	27	0	0	1	12.0000	12		
29	26	26	5	26	0	0	2	12.0000		11	
30	33	33	7	33	0	0	3	13.0000			7
31	32	32	5	32	0	0	2	13.0000		12	
32	31	31	5	31	0	0	2	13.0000		13	
33	30	30	6	30	0	0	3	13.0000			8
34	36	36	7	36	0	0	3	14.0000			9
35	35	35	5	35	0	0	2	14.0000		14	
36	34	34	6	34	0	0	3	14.0000			10
37	37	37	6	37	0	0	3	15.0000			11
38	39	39	6	39	0	0	3	16.0000			12
39	38	38	6	38	0	0	3	16.0000			13

TABLE 3. TRADEOFF FOR FEATURE NUMBER 2

DATE	610	ID	TC1AM1	NUMBER	2	CL	SCORE	0	1	2	3
TRADEOFF	N0	ID	TY								
1	2	2	4	2	0	0	1	3.0000	1		
2	30	30	6	30	0	0	3	4.0000			1
3	16	16	4	16	0	0	1	4.0000	2	1	
4	3	3	4	3	0	0	1	4.0000	3		
5	1	1	4	1	0	0	1	4.0000	4		
6	38	38	6	38	0	0	3	5.0000			2
7	34	34	6	34	0	0	3	5.0000			3
8	21	21	6	21	0	0	3	5.0000			4
9	11	11	4	11	0	0	1	5.0000	5		
10	5	5	4	5	0	0	1	5.0000	6		
11	4	4	4	4	0	0	1	5.0000	7		
12	39	39	6	39	0	0	3	6.0000			5
13	37	37	6	37	0	0	3	6.0000			6
14	17	17	4	17	0	0	1	6.0000	8		
15	26	26	5	26	0	0	2	7.0000		1	
16	10	10	4	10	0	0	1	7.0000	9		
17	7	7	5	7	0	0	2	7.0000		2	
18	31	31	5	31	0	0	2	8.0000		3	
19	27	27	4	27	0	0	1	8.0000	10		
20	18	18	5	18	0	0	2	8.0000		4	
21	12	12	5	12	0	0	2	8.0000		5	
22	35	35	5	35	0	0	2	9.0000		6	
23	22	22	5	22	0	0	2	9.0000		7	
24	13	13	5	13	0	0	2	9.0000		8	
25	6	6	4	6	0	0	1	9.0000	11		
26	32	32	5	32	0	0	2	10.0000		9	
27	23	23	5	23	0	0	2	10.0000		10	
28	9	9	5	9	0	0	2	10.0000		11	
29	28	28	5	28	0	0	2	11.0000		12	
30	19	19	5	19	0	0	2	11.0000		13	
31	14	14	5	14	0	0	2	11.0000		14	
32	33	33	7	33	0	0	3	13.0000			7
33	20	20	7	20	0	0	3	13.0000			8
34	8	8	4	8	0	0	1	13.0000	12		
35	29	29	7	29	0	0	3	14.0000			9
36	24	24	7	24	0	0	3	14.0000			10
37	25	25	7	25	0	0	3	15.0000			11
38	15	15	7	15	0	0	3	15.0000			12
39	36	36	7	36	0	0	3	16.0000			13

and

$$r_{1,2}^{-1} = \left( \frac{\Delta m_1}{\tilde{\sigma}_1} \right)^2 + \left( \frac{\Delta m_2}{\tilde{\sigma}_2} \right)^2 = 4.07$$

- $j = 1, k = 3$  -  $\Delta m_1^2 = (\hat{m}_{1,1} - \hat{m}_{3,1})^2 = (6.417 - 12.62)^2 = 38.48$   
 $\Delta m_2^2 = (\hat{m}_{1,2} - \hat{m}_{3,2})^2 = (6.083 - 10.08)^2 = 16$   
 $\tilde{\sigma}_1^2 = (12\hat{\sigma}_{1,1}^2 + 13\hat{\sigma}_{3,1}^2)/25 = \frac{(12 \cdot 7.076 + 13 \cdot 5.314)}{25}$   
 $= 6.16$   
 $\tilde{\sigma}_2^2 = (12\hat{\sigma}_{1,2}^2 + 13\hat{\sigma}_{3,2}^2)/25 = \frac{(12 \cdot 7.243 + 13 \cdot 21.46)}{25}$   
 $= 14.64$

and

$$r_{1,3}^{-1} = \frac{38.48}{6.16} + \frac{16}{14.64} = 7.34$$

- $j = 2, k = 3$   $\Delta m_1^2 = (\hat{m}_{2,1} - \hat{m}_{3,1})^2 = 6.5$   
 $\Delta m_2^2 = (\hat{m}_{2,2} - \hat{m}_{3,2})^2 = 0.88$   
 $\tilde{\sigma}_1^2 = (14\hat{\sigma}_{2,1}^2 + 13\hat{\sigma}_{3,1}^2)/27 = 6.15$   
 $\tilde{\sigma}_2^2 = (14\hat{\sigma}_{2,2}^2 + 13\hat{\sigma}_{3,2}^2)/27 = 11.29$

and

$$r_{2,3}^{-1} = 1.13.$$

From (21), we can expect a larger error rate for decisions between classes  $C_2$  and  $C_3$ , since the inter-class complexity for  $C_2$  and  $C_3$  is much higher (at least twice) than for class pairs  $C_1$  and  $C_2$  and  $C_1$  and  $C_3$ . This result

tallies with an 'eyeball' examination of Figure 6 where one can see that class  $C_3$  brackets class  $C_2$ .

Intra-class shape complexity and orientation is determined using the results from step 2. The values of the cross-correlation coefficients  $\rho_{12}(i)$  and the standard deviations  $\hat{\sigma}_{i,1}, \hat{\sigma}_{i,2}$  are enough to determine a closed contour. For each class,  $i = 1, 2, 3$  such that, if the Unimodal Gaussian (UG) assumption holds, then

- 1)  $p(X|C_i) = 0.6065/K$  for each  $X$  on the contour,  $K = [(2\pi)^n \det \hat{V}_i]^{-1/2}$ .
- 2) The contour encloses, and is centered upon  $\hat{m}_i$ , and is ellipsoidal in shape.
- 3) Letting  $X_i$  denote the set of points interior to the contour for  $C_i$ , we have  $P(X \in X_i | C_i) = 0.3935$ . This probability can be determined for any  $n$  using a transformation to spherical coordinates and integration by parts.

Since

$$\begin{bmatrix} \hat{\sigma}_{i,1}^2 & \rho_{12}(i) \hat{\sigma}_{i,1} \hat{\sigma}_{i,2} \\ \rho_{12}(i) \hat{\sigma}_{i,1} \hat{\sigma}_{i,2} & \hat{\sigma}_{i,2}^2 \end{bmatrix} = \hat{V}_i$$

these values are readily available for each  $C_i$ . Properties 1 and 3 ensure that the shape of each contour (property 2) is similar to the shapes for other classes.

To determine the shape of the  $i^{\text{th}}$  class under the UG assumptions, we determine  $\hat{m}_i$  and the four points  $(\hat{m}_{i,1} \pm \hat{\sigma}_{i,1}, \hat{m}_{i,2})$ ,  $(\hat{m}_{i,1}, \hat{m}_{i,2} \pm \hat{\sigma}_{i,2})$ , as shown in Figure 7a. These points are on the ellipsoidal contour with center



$\hat{m}_i$  and having a major axis of slope  $\rho_{12}(i)$  through  $\hat{m}_i$ . Figure 7b shows the contours for each of  $C_1$ ,  $C_2$ , and  $C_3$ . (Note the similarity between contour plotting and scatter plotting of the data.)

From Figure 7b, we can see that there is extensive overlap between classes  $C_2$  and  $C_3$ , and that  $C_3$  is widely dispersed compared to  $C_1$  and  $C_2$ , since each of the contours shown is equi-probable (property 3). This large dispersion is an indicator of a multi-modal class, but is not conclusive.

If we did not check modality, the results of computations 1 and 2, given by equation (21) and Figure 7b tell us that classes  $C_2$  and  $C_3$  cannot be reliably separated by a linear classifier on features 1 and 2 (i.e. using the UG assumptions will not lead to an acceptable classifier). To show this, we design the linear classifier for  $C_1$ ,  $C_2$ , and  $C_3$  as follows:

From Appendix A, equation (A-9), we have the expression for the two-class discriminant function  $\hat{\Lambda}_2^L(X)$ . We use this for each of the class pairs  $C_1C_2$ ,  $C_1C_3$ , and  $C_2C_3$  to get the decision rules,  $(i,j) = (1,2)$ ,  $(1,3)$ , and  $(2,3)$ ,

$$\Lambda_{ij}(X) = (X^T - \rho_j^j \hat{m}_i^T - \rho_j^i \hat{m}_j^T) \hat{V}_{ij}^{-1} (\hat{m}_j - \hat{m}_i) - K_L(i,j) \begin{matrix} C_j \\ > \\ C_i \end{matrix} 0. \quad (22)$$

Using equations (A-7) and (A-8), we have the equivalent form

$$\Lambda_{ij}(X) = X^T \cdot \vec{W}_{i,j} - W_{0,i,j} \begin{matrix} C_j \\ > \\ C_i \end{matrix} 0 \quad (23)$$

where

$$\begin{aligned} (W_{1,i,j}, W_{2,i,j}) &= \hat{V}_{i,j}^{-1} (\hat{m}_j - \hat{m}_i) \\ W_{0,i,j} &= K_L(i,j) + (\rho_i^j \hat{m}_i + \rho_j^i \hat{m}_j)^T \hat{V}_{i,j}^{-1} (\hat{m}_j - \hat{m}_i) \end{aligned} \quad (24)$$

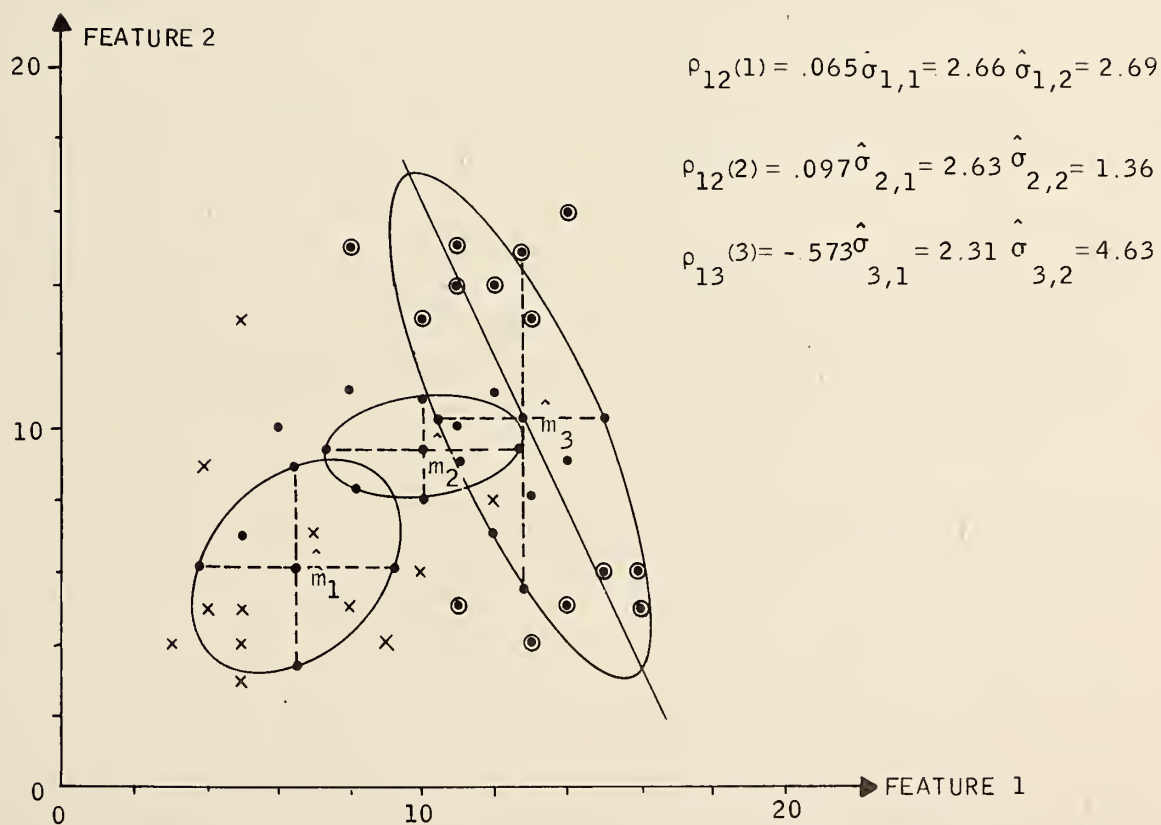
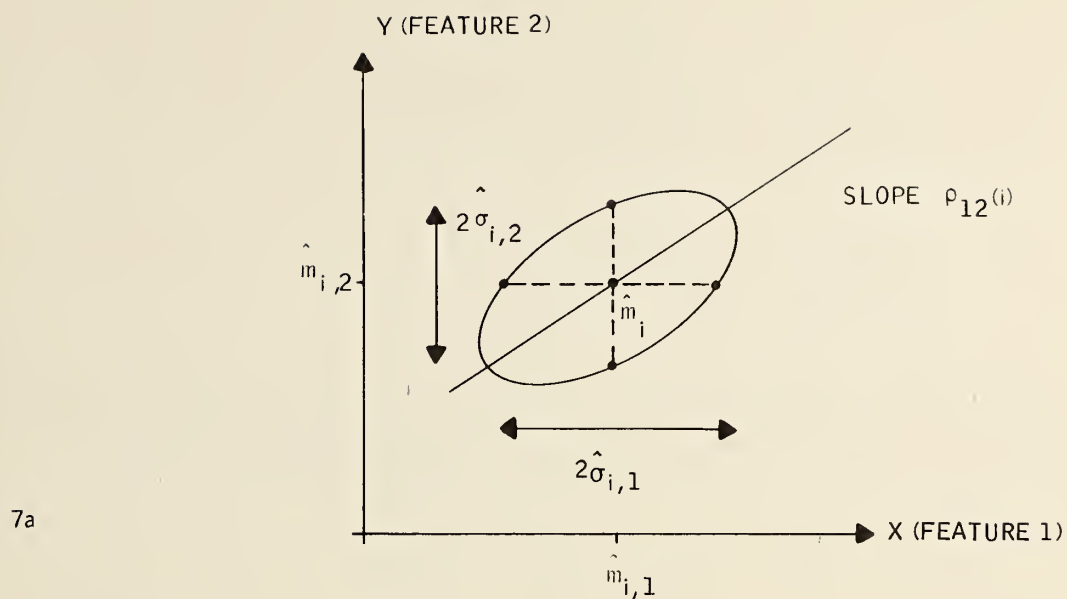


Figure 7. Contour Approximation to Distribution Shape

and

$$\rho_i^j = \frac{M_i}{M_i + M_j} = \frac{P(C_i)}{P(C_i) + P(C_j)}$$

$$\rho_j^i = \frac{M_j}{M_i + M_j} = \frac{P(C_j)}{P(C_i) + P(C_j)}.$$

The CTC program determines the decision surface using the form (j>i) (the reason for choosing equation (25) is explained in Chapter 2, and in Appendix A,

$$\Lambda_{ij}^{CTC}(X) = [X^T - (\rho_j^i \hat{m}_j + \rho_i^j \hat{m}_i)^T] \hat{V}_{ij}^{-1} (\hat{m}_j - \hat{m}_i) (j - i) \rho_i \rho_j$$

$$+ \begin{matrix} C_j \\ i \rho_i^j + j \rho_j^i > \\ C_i < \end{matrix} \text{const}(i, j) \quad (25)$$

where  $\text{const}(i, j)$  is determined so as to minimize the error rate on the samples, by using a property listing of the classifier output (score) determined by computing  $\hat{\Lambda}_{ij}^{CTC}(X)$ , as given on the left side of the inequality (25). This can be seen equivalent to equation 22 by defining

$$-K_L(i, j) = \frac{(i \rho_i^j + j \rho_j^i)}{(j - i) \rho_i^j \rho_j^i} - \frac{\text{const}(i, j)}{(j - i) \rho_i^j \rho_j^i} \quad (26)$$

and taking

$$\Lambda_{ij}(X) = \Lambda_{ij}^{CTC}(X) / [j - i] \rho_i^j \rho_j^i.$$

The resulting coefficients  $W_{k, i, j}^{CTC}$ ,  $k = 0, 1, 2$ , the  $(i, j)$  pairs as before, are computed by the CTC program as in Table 4, and are

(i, j)	$\tilde{W}_{0,i,j}^{CTC}$	$W_{1,i,j}^{CTC}$	$W_{2,i,j}^{CTC}$
1, 2	0.3648	0.06199	0.0846
1, 3	-0.3233	0.1810	0.07582
2, 3	1.076	0.09239	0.03776

where, using (25)

$$\Lambda_{ij}^{CTC}(X) = \bar{X} \cdot (W_{1,i,j}^{CTC}, W_{2,i,j}^{CTC}) - W_{0,i,j}^{CTC} \begin{matrix} C_j \\ > 0 \\ C_i \\ < 0 \end{matrix} \quad (27)$$

with

$$\bar{W}_{ij}^{CTC} = (W_{1,i,j}^{CTC}, W_{2,i,j}^{CTC})^T = \hat{V}_{i,j}^{-1} (\hat{m}_j - \hat{m}_i) (j - i) \rho_i^j \rho_j^i \quad (28)$$

and

$$W_{0,i,j}^{CTC} = \text{const}(i, j) + \overbrace{(\rho_i^j \hat{m}_i + \rho_j^i \hat{m}_j) \cdot \bar{W}_{i,j}^{CTC} - i \rho_i^j - j \rho_j^i}^{\tilde{W}_{0,i,j}^{CTC}}.$$

The values for  $\text{const}(1, 2)$ ,  $\text{const}(1, 3)$ , and  $\text{const}(2, 3)$  are determined from the property listings given by Tables 5 through 7). The 'SCORE' column gives the value, for each sample point  $X_r$ ,  $r = 1, \dots, 39$ , (see equation (28) for  $W_{0,i,j}^{CTC}$ )



TABLE 4. PAIRWISE CLASSIFIER COEFFICIENTS

METHOD 1

INVERSE

K1 = 1

K2 = 1

DET = 58.555530294

1	2	.3648	.6199E-01	.8460E-01
---	---	-------	-----------	-----------

INVERSE

K1 = 1

K2 = 1

DET = 282.83129606

1	3	-.3233	.1810	.7582E-01
---	---	--------	-------	-----------

INVERSE

K1 = 1

K2 = 1

DET = 84.558857982

2	3	1.076	.9239E-01	.3776E-01
---	---	-------	-----------	-----------

TABLE 5. TRADEOFF FOR PAIR 1, 2

DATE 615 ID TC1--TI  
TRADEOFF FOR  
PAIR 1 2

COEFF

•36483		•06198		•08459		CL
NR	ID TY	ID TY	ID TY	ID TY	ID TY	
1	1	1	1	0	0	0 1
2	2	2	1	0	0	0 1
3	3	3	1	0	0	0 1
4	5	5	1	0	0	0 1
5	4	4	1	0	0	0 1
6	16	16	1	0	0	0 1
7	7	7	2	0	0	0 2
8	11	11	1	0	0	0 1
9	6	6	1	0	0	0 1
10	10	10	1	0	0	0 1
11	21	21	3	0	0	0 3
12	17	17	1	0	0	0 1
13	30	30	3	0	0	0 3
14	12	12	2	0	0	0 2
15	9	9	2	0	0	0 2
16	13	13	2	0	0	0 2
17	34	34	3	0	0	0 3
18	18	18	2	0	0	0 2
19	26	26	2	0	0	0 2
20	8	8	1	0	0	0 1
21	38	38	3	0	0	0 3
22	27	27	1	0	0	0 1
23	14	14	2	0	0	0 2
24	37	37	3	0	0	0 3
25	22	22	2	0	0	0 2
26	31	31	2	0	0	0 2
27	39	39	3	0	0	0 3
28	23	23	2	0	0	0 2
29	19	19	2	0	0	0 2
30	35	35	2	0	0	0 2
31	32	32	2	0	0	0 2
32	28	28	2	0	0	0 2
33	20	20	4	0	0	0 4
34	15	15	4	0	0	0 4
35	24	24	4	0	0	0 4
36	33	33	4	0	0	0 4
37	29	29	4	0	0	0 4
38	25	25	4	0	0	0 4
39	36	36	4	0	0	0 4

SCORE	0	1	2	3
•8891		1		3
•9285		2		
1.0131		3		
1.0357		4		
1.0977		5		
1.2617		6	error 1	
1.2669		7	①	
1.2836		8		
1.3741	5	9		
1.3908	4			
1.4696				1
1.4922	3	1		
1.5374			2	
1.5826	4			
1.6220	5			
1.6555				3
1.6614			5	
1.7007		error 2	6	
1.7744		⑪		
1.7795				4
1.7853		⑫		
1.7912		error 3	7	
1.8321				5
1.8579			8	
1.8473			9	
1.8641				6
1.8925			10	
1.9151			11	
1.9339			12	
2.0165			13	
2.0391			14	
2.0643				7
2.1295				8
2.2309				9
2.2703				10
2.2929				11
2.3155				12
2.5260				13

3 errors minimum  
const (1, 2) =  
 $1.4922 + \frac{(1.5374 - 1.4922)}{2}$   
= 1.5148

TABLE 6. TRADEOFF FOR PAIR 1, 3

DATE 615 ID TC1--TT  
TRADEOFF FOR  
PAIR 1 3

COEFF

••32328

•18098

•C7582

NO	ID	TY	CL	SCORE	0	1	2	3
1	1	1	0	0	1			
2	5	5	1	0	1			
3	2	2	1	0	1			
4	3	3	1	0	1			
5	4	4	1	0	1			
6	6	6	1	0	1			
7	7	7	2	0	0			
8	10	10	1	0	0			
9	11	11	1	0	0			
10	9	9	2	0	0			
11	8	8	1	0	0			
12	16	16	1	0	0			
13	12	12	2	0	0			
14	13	13	2	0	0			
15	17	17	1	0	0			
16	14	14	2	0	0			
17	21	21	3	0	0			
18	18	18	2	0	0			
19	15	15	4	0	0			
20	19	19	2	0	0			
21	30	30	3	0	0			
22	22	22	2	0	0			
23	26	26	2	0	0			
24	23	23	2	0	0			
25	27	27	1	0	0			
26	20	20	4	0	0			
27	34	34	3	0	0			
28	31	31	2	0	0			
29	28	28	2	0	0			
30	24	24	4	0	0			
31	32	32	2	0	0			
32	25	25	4	0	0			
33	37	37	3	0	0			
34	35	35	2	0	0			
35	29	29	4	0	0			
36	38	38	3	0	0			
37	33	33	4	0	0			
38	39	39	3	0	0			
39	36	36	4	0	0			

SCORE	0	1	2	3
•5229		1		
•7797		2		
•8191		3		
•8849		4		
•9607		5		
1.0230		6		
1.1123			1	
1.4743		7		
1.5136		8		
1.5207			2	
1.5672		9		
1.6188		10		
1.7311			3	
1.8169			4	
1.9414	11			
1.9585			5	
2.0465				1
2.0930			6	
2.2618				2
2.3225			7	
2.3327				3
2.3498			8	
2.3792			9	
2.4156	error 1	10		
2.4550	(12)			
2.4721			4	
2.5205			5	
2.6359		11		
2.6524		12		
2.7289			6	
2.7276		13		
2.8147			7	
2.8463			8	
2.8927		14		
2.9199			9	
2.9514			10	
3.0150			11	
3.0272			12	
3.4235			13	

1 error minimum

const (1, 3) =

$$1.9414 + \frac{(2.0465 - 1.9414)}{2} = 1.9940$$

TABLE 7. TRADEOFF FOR PAIR 2, 3

DATE 615 ID TC1--TT  
TRADEOFF FOR  
PAIR 2 3

COEFF

1.07573	.09238	.03775	CL	SCORE	1	2	3
18	ID TY						
1 1	1 1	0 0	0 1	1.5139	1		
2 5	5 1	0 0	0 1	1.6241	2		
3 2	2 1	0 0	0 1	1.6519	3		
4 3	3 1	0 0	0 1	1.6887	4		
5 4	4 1	0 0	0 1	1.7264	5		
6 6	6 1	0 0	0 1	1.7851	6		
7 7	7 2	0 0	0 2	1.8119		1	
8 10	10 1	0 0	0 1	1.9867	7		
9 11	11 1	0 0	0 1	2.0136	3		
10 9	9 2	0 0	0 2	2.0176		2	
11 8	8 1	0 0	0 1	2.0285	3		
12 16	16 1	0 0	0 1	2.0582	10		
13 12	12 2	0 0	0 2	2.1168		3	
14 13	13 2	0 0	0 2	2.1546		4	
15 17	17 1	0 0	0 1	2.2261	11		
16 14	14 2	0 0	0 2	2.2301		5	
17 21	21 3	0 0	0 3	2.2807		1	9
18 18	18 2	0 0	0 2	2.316		6	10
19 15	15 4	0 0	0 3	2.3811		2	9
20 19	19 2	0 0	0 2	2.4149		7	10
21 30	30 3	0 0	0 3	2.4277		3	9
22 22	22 2	0 0	0 2	2.4317		8	10
23 26	26 2	0 0	0 2	2.4486		9	9
24 23	23 2	0 0	0 2	2.4695		10	8
25 27	27 1	0 0	0 1	2.4864	12		7
26 20	20 4	0 0	0 3	2.4904		4	8
27 34	34 3	0 0	0 3	2.579		5	9
28 31	31 2	0 0	0 2	2.5787	11		8
29 28	28 2	0 0	0 2	2.5996	12		7
30 24	24 4	0 0	0 3	2.6215		6	8
31 32	32 2	0 0	0 2	2.6543	12		7
32 25	25 4	0 0	0 3	2.6583		7	8
33 37	37 3	0 0	0 3	2.6880		8	9
34 35	35 2	0 0	0 2	2.7089	14		8
35 29	29 4	0 0	0 3	2.7129		9	
36 38	38 3	0 0	0 3	2.7426		10	
37 33	33 4	0 0	0 3	2.7675		11	
38 39	39 3	0 0	0 3	2.7804		12	
39 36	36 4	0 0	0 3	2.9731		13	

7 errors minimum,  
no unique choice for  
const (2, 3)

$$\frac{2.4695 + 2.4904}{2} = 2.47995$$

$$\frac{2.5996 + 2.6205}{2} = 2.61005$$

$$\frac{2.6543 + 2.6583}{2} = 2.6563$$



$$\text{score}(r) = X_r \cdot \underbrace{(W_{1,i,j}^{\text{CTC}}, W_{2,i,j}^{\text{CTC}})}_{\vec{W}_{i,j}^{\text{CTC}}} - (\rho_i^j \hat{m}_i + \rho_j^i \hat{m}_j) \cdot \bar{W}_{i,j}^{\text{CTC}} - i\rho_i^j - j\rho_j^i.$$

Thus,

$$i = 1, j = 2: (2.2 - 14) \Rightarrow \text{const}(1, 2) = 1.5148 \quad (3 \text{ errors})$$

$$i = 1, j = 3: (2.2 - 15) \Rightarrow \text{const}(1, 3) = 1.9940 \quad (1 \text{ error})$$

$$i = 2, j = 3: (2.2 - 16) \Rightarrow \text{const}(2, 3) = \begin{cases} 2.47995 & (7 \text{ errors}) \\ 2.61005 & (7 \text{ errors}) \\ 2.6563 & (7 \text{ errors}) . \end{cases}$$

Note that ratios of the interclass complexity  $r_{12}/r_{13}$ , etc. (equation 21) provide a coarse estimate of ratios of the above error rates.

Further note that our troublesome  $C_2 C_3$  pair has no unique solution for the minimum error rate of 7.

What is the source of our trouble with  $C_2 C_3$ ? (It is obvious when the data are plotted as in Figure 6, but we are designing the classifier as if such a plot were not available.)

Let us summarize the results obtained so far:

- i)  $r_{2,3}$  is large (interclass complexity) relative to other class pairs
- ii)  $C_3$  is widely dispersed throughout  $X_f$  compared to other classes

- iii) the best linear discriminant  $\Lambda_{2,3}^{CTC}(X)$  is not unique and the minimum error rate on the samples is high, compared to other class-pair decision rules, each of which has a unique low error rate.

Each of i, ii, and iii is a characteristic of a problem for which one of the classes in a 'difficult' class pair is multimodal with the modes arranged so that the difficult pair is not linearly separable.

Note that since the classes  $C_1$  and  $C_3$  are well separated using a linear decision rule, this shows that multimodality in itself is not necessarily a problem ( $C_3$  is bimodal  $C_1$  is not). Only when the modes lie on either side of another class do the problems arise in using the UG assumptions to design a classifier.

To check modality for this three-class problem, since i, ii, and iii indicate the possible presence of a multimodal class with the modes arranged in  $X_F$  so as to 'bracket' another class, we use the property listings of Table 2 and 3. Table 2 does not indicate multimodality for any of the classes (projection onto Feature 1). Table 3 tells the story: note the two clusters of sample points for class  $C_3$ . This property listing on Feature 2, together with results i, ii, and iii, provides compelling evidence that  $C_3$  is multimodal in such a way that the UG assumptions cannot be used to design a reliable linear classifier.

### 3.3.2 Piecewise Linear Solution Using Subclasses

We will next extend the multimodal gaussian (MG) problem to a four-class UG problem. From the results of the preceding section, we know that  $C_3$  is bimodal, with the sample points arranged in two modes as (from Table 3):

Mode 1:  $\{X_r: r = 30, 38, 34, 21, 39, 37\}$  (type 6)

Mode 2:  $\{X_r: r = 33, 20, 29, 24, 25, 15, 36\}$  (type 7).

If the property listings on each feature are not sufficient to group data points into separate modes as done above, other projections must be used for property listing. Some useful projections are described in Appendix C for such cases. Using these data types, the extended four class problem is attacked using the type to class conversion.

Type	Class
4	1
5	2
6	3
7	4

As before, the computations required for complexity analysis and classifier design give the following results:

Step 0. A priori probabilities (class weights);  $M = 39$

$i$	$M_i$	$P(C_i) = \frac{M_i}{M}$
1	12	0.31
2	14	0.36
3	6	0.1538
4	7	0.1795

Step 1. Distance computations

$$\hat{m}_1 = (6.417, 6.083)$$

$$\hat{m}_2 = (10.07, 9.143)$$

$$\hat{m}_3 = (14.17, 5.167)$$

$$\hat{m}_4 = (11.29, 14.29)$$

$$d(\hat{m}_i, \hat{m}_j) = \begin{cases} 4.766 & , \quad i=1, j=2 \\ 7.804 & , \quad i=1, j=3 \\ 9.539 & , \quad i=1, j=4 \\ 5.708 & , \quad i=2, j=3 \\ 5.284 & , \quad i=2, j=4 \\ 9.563 & , \quad i=3, j=4 \end{cases}$$

$$\theta_{ij} = \begin{cases} 149.4^\circ & , \quad i=1, j=2 \\ 106.8^\circ & , \quad i=1, j=3 \\ 140.2^\circ & , \quad i=1, j=4 \\ 103.7^\circ & , \quad i=2, j=3 \\ 9.219^\circ & , \quad i=2, j=4 \\ 113.0^\circ & , \quad i=3, j=4 \end{cases}$$

Step 2. Statistical scatter computation

$$\hat{V}_1 = \begin{bmatrix} 7.076 & 0.4653 \\ 0.4653 & 7.243 \end{bmatrix} , \quad \rho_{12}^{(1)} = 0.06499$$

$$\hat{V}_2 = \begin{bmatrix} 6.923 & 0.3469 \\ 0.3469 & 1.837 \end{bmatrix} , \quad \rho_{12}^{(2)} = 0.09729$$

$$\hat{V}_3 = \begin{bmatrix} 3.139 & 0.6389 \\ 0.6389 & 0.4722 \end{bmatrix} , \quad \rho_{12}^{(3)} = 0.5248$$



$$\hat{V}_4 = \begin{bmatrix} 3.347 & 0.2041 \\ 0.2041 & 1.061 \end{bmatrix}, \quad \rho_{12}^{(4)} = 0.1083$$

Step 3. Property listing - on Feature 1 in Table 8  
 - on Feature 2 in Table 9.

Since we have extended the original three-class problem, dimensionality must be checked again:

$$n = 2, \quad N = 4, \quad M_i \geq 6$$

hence

$$\frac{M_i}{n} \geq 3, \quad N-1 \not\geq n,$$

and the  $\frac{M_i}{n}$  constraint is barely satisfied, while the  $\frac{N}{n}$  constraint  $\frac{N}{n} \geq \frac{1}{1-\frac{1}{N}}$  is not satisfied.

We will continue with the solution in spite of this, because dimensionality constraints were satisfied for the original three-class problem. Since the statistical representativeness of the extended problem is not clear, the classifier will have to be designed by making use of high sample density regions in feature space when ambiguous cases arise [i. e., multiple minima when choosing  $\text{const}(i, j)$ ].

TABLE 8. TRADEOFF FOR FEATURE NUMBER 1

DATE	610	ID	TC1BM1	NUMBER	1	CL	SCORE	0	1	2	3	4
TRADEOFF	N0	ID	TY									
1	1	1	4	1	0	0	3.0000		1			
2	6	6	4	6	0	0	4.0000		2			
3	5	5	4	5	0	0	4.0000		3			
4	8	8	4	8	0	0	5.0000		4			
5	7	7	5	7	0	0	5.0000			1		
6	4	4	4	4	0	0	5.0000		5			
7	3	3	4	3	0	0	5.0000		6			
8	2	2	4	2	0	0	5.0000		7			
9	9	9	5	9	0	0	6.0000			2		
10	10	10	4	10	0	0	7.0000		8			
11	15	15	7	15	0	0	8.0000					1
12	14	14	5	14	0	0	8.0000			3		
13	13	13	5	13	0	0	8.0000			4		
14	12	12	5	12	0	0	8.0000			5		
15	11	11	4	11	0	0	8.0000		9			
16	16	16	4	16	0	0	9.0000		10			
17	20	20	7	20	0	0	10.0000					2
18	19	19	5	19	0	0	10.0000			6		
19	18	18	5	18	0	0	10.0000			7		
20	17	17	4	17	0	0	10.0000		11			
21	25	25	7	25	0	0	11.0000					3
22	24	24	7	24	0	0	11.0000					4
23	23	23	5	23	0	0	11.0000			8		
24	22	22	5	22	0	0	11.0000			9		
25	21	21	6	21	0	0	11.0000				1	
26	29	29	7	29	0	0	12.0000					5
27	28	28	5	28	0	0	12.0000			10		
28	27	27	4	27	0	0	12.0000		12			
29	26	26	5	26	0	0	12.0000			11		
30	33	33	7	33	0	0	13.0000					6
31	32	32	5	32	0	0	13.0000			12		
32	31	31	5	31	0	0	13.0000			13		
33	30	30	6	30	0	0	13.0000				2	
34	36	36	7	36	0	0	14.0000					7
35	35	35	5	35	0	0	14.0000			14		
36	34	34	6	34	0	0	14.0000				3	
37	37	37	6	37	0	0	15.0000				4	
38	39	39	6	39	0	0	16.0000				5	
39	38	38	6	38	0	0	16.0000				6	

TABLE 9. TRADEOFF FOR FEATURE NUMBER 2

DATE	610	ID	TC13M1	TRADEOFF FOR FEATURE	NUMBER	2	CL	SCORE	0	1	2	3	4
	NB	ID	TY										
1	2	2	4	2	0	0	1	3.0000		1			
2	30	30	6	30	0	0	3	4.0000				1	
3	16	16	4	16	0	0	1	4.0000		2			
4	3	3	4	3	0	0	1	4.0000		3			
5	1	1	4	1	0	0	1	4.0000		4			
6	38	38	6	38	0	0	3	5.0000				2	
7	34	34	6	34	0	0	3	5.0000				3	
8	21	21	6	21	0	0	3	5.0000				4	
9	11	11	4	11	0	0	1	5.0000		5			
10	5	5	4	5	0	0	1	5.0000		6			
11	4	4	4	4	0	0	1	5.0000		7			
12	39	39	6	39	0	0	3	6.0000				5	
13	37	37	6	37	0	0	3	6.0000				6	
14	17	17	4	17	0	0	1	6.0000		8			
15	26	26	5	26	0	0	2	7.0000			1		
16	10	10	4	10	0	0	1	7.0000		9			
17	7	7	5	7	0	0	2	7.0000			2		
18	31	31	5	31	0	0	2	8.0000			3		
19	27	27	4	27	0	0	1	8.0000		10			
20	18	18	5	18	0	0	2	8.0000			4		
21	12	12	5	12	0	0	2	8.0000			5		
22	35	35	5	35	0	0	2	9.0000			6		
23	22	22	5	22	0	0	2	9.0000			7		
24	13	13	5	13	0	0	2	9.0000			8		
25	6	6	4	6	0	0	1	9.0000		11			
26	32	32	5	32	0	0	2	10.0000			9		
27	23	23	5	23	0	0	2	10.0000			10		
28	9	9	5	9	0	0	2	10.0000			11		
29	28	28	5	28	0	0	2	11.0000			12		
30	19	19	5	19	0	0	2	11.0000			13		
31	14	14	5	14	0	0	2	11.0000			14		
32	33	33	7	33	0	0	4	13.0000					1
33	20	20	7	20	0	0	4	13.0000					2
34	8	8	4	8	0	0	1	13.0000		12			
35	29	29	7	29	0	0	4	14.0000					3
36	24	24	7	24	0	0	4	14.0000					4
37	25	25	7	25	0	0	4	15.0000					5
38	15	15	7	15	0	0	4	15.0000					6
39	36	36	7	36	0	0	4	16.0000					7

The inter-class complexity measure is

$$r_{jk} = \begin{cases} 0.25 & , \quad j=1, k=2 \\ 0.09 & , \quad j=1, k=3 \\ 0.06 & , \quad j=1, k=4 \\ 0.07 & , \quad j=2, k=3 \\ 0.06 & , \quad j=2, k=4 \\ 0.03 & , \quad j=3, k=4 \end{cases} \quad (29)$$

where, from Equation (20),

$$\tilde{\sigma}_i^2 = \left[ \frac{\rho_j^k \hat{\sigma}_{j,i}^2 + \rho_k^j \hat{\sigma}_{k,i}^2}{\rho_j^k + \rho_k^j} \right] = \left[ \frac{M_j \hat{\sigma}_{j,i}^2 + M_k \hat{\sigma}_{k,i}^2}{M_j + M_k} \right]$$

(i, j)	$\Delta m_1^2$	$\Delta m_2^2$	$\tilde{\sigma}_1^2$	$\tilde{\sigma}_2^2$	$r_{ij}^{-1}$
(1, 2)	13.34	9.36	6.99	4.33	4.07
(1, 3)	60.11	0.84	5.76	4.99	10.60
(1, 4)	23.75	67.35	5.70	4.97	17.72
(2, 3)	16.81	15.81	5.79	1.43	13.96
(2, 4)	1.49	26.49	5.73	1.53	17.57
(3, 4)	8.29	83.23	2.99	0.79	30.61

Intra-class shape complexity is plotted in Figure 3 using

i	$\hat{\sigma}_{i1}$	$\hat{\sigma}_{i2}$	$\rho_{12}(i)$
1	2.66	2.69	0.065
2	2.63	1.36	0.097
3	1.77	0.69	0.525
4	1.83	1.03	0.108

and the points  $\hat{m}_i$  from computation 1.



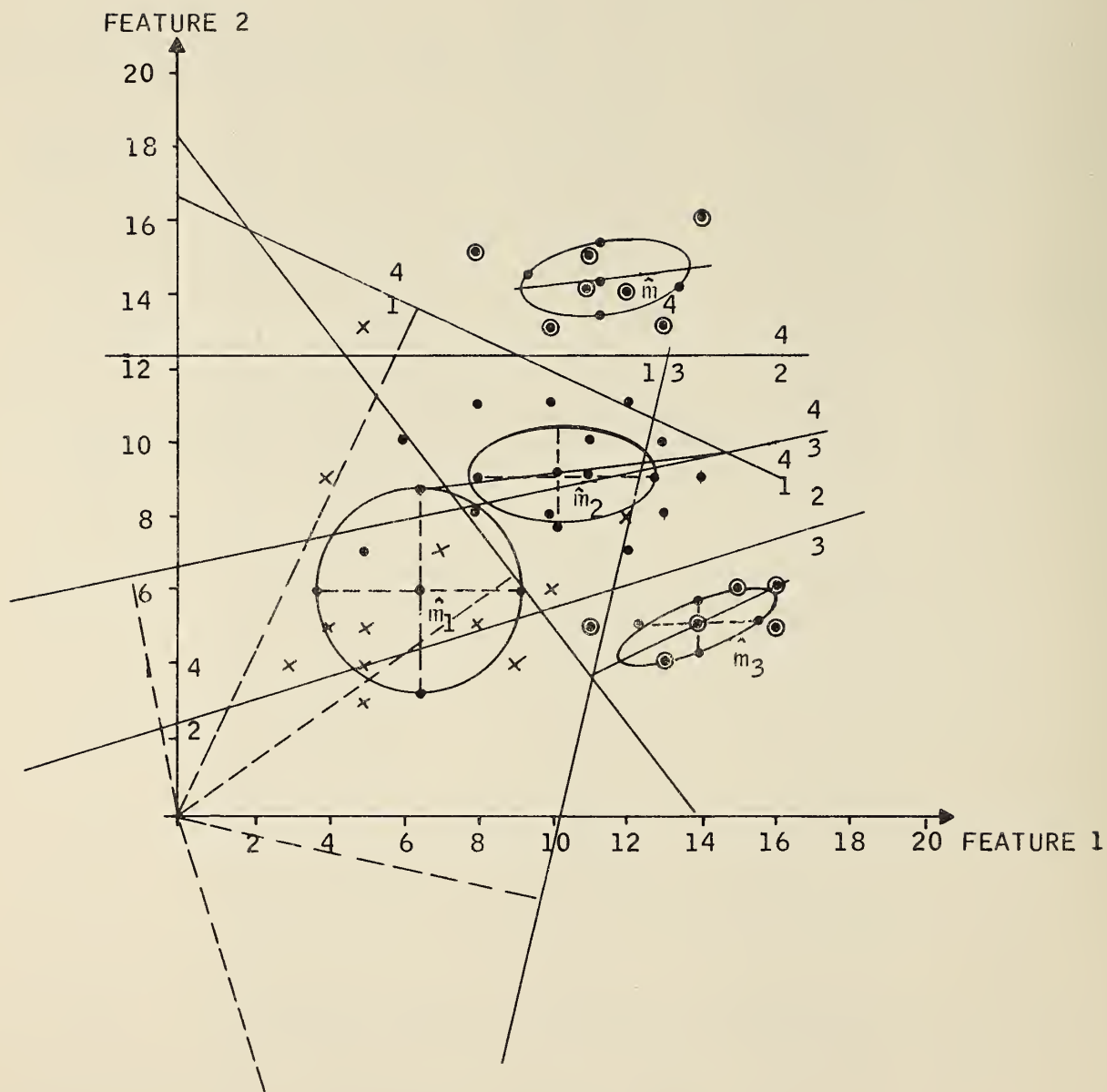


Figure 8. Intra-class Shape Complexity Plot

Finally, no class shows multimodality on the property listings on Feature 1 and 2 in Tables 8 and 9.

From the foregoing results it can be seen that the four-class problem fits the UG assumptions quite well, hence the original three-class problem is a three-class MG problem with a piecewise-linear discriminant classifier solution. From Appendix B, we know that the linear classifier solution to the four-class UG problem gives the piecewise-linear solution to the three-class MG problem. The CTC program solves the four-class problem, using linear discriminants, as follows: (see program output in Table 10).

(i, j)	$-\tilde{W}_{0ij}^{CTC}$	$W_{1ij}^{CTC}$	$W_{2ij}^{CTC}$
1, 2	0.3648	0.06199	0.08460
1, 3	0.311	0.1779	-0.04246
1, 4	-0.845	0.1070	0.2275
2, 3	2.964	0.04507	-0.1475
2, 4	-0.6936	-0.00396	0.3057
3, 4	2.774	-0.01938	0.1001

and from Tables 11 through 16,

(i, j)	const(i, j)	#errors
1, 2	1.51	3
1, 3	2.144	2
1, 4	2.92	0
2, 3	2.60	0
2, 4	3.02	0
3, 4	3.45	0

TABLE 10. PAIRWISE CLASSIFIER COEFFICIENTS

	140.2	9.219	113.0	.0000
--	-------	-------	-------	-------

METHOD 1

INVERSE  
 $K_1 = 1$   
 $K_2 = 1$   
DET = 58.555530294

1	2	.3648	.6199E-01	.8460E-01
---	---	-------	-----------	-----------

INVERSE  
 $K_1 = 1$   
 $K_2 = 1$   
DET = 97.744513062

1	3	.3110	.1779	-.4246E-01
---	---	-------	-------	------------

INVERSE  
 $K_1 = 1$   
 $K_2 = 1$   
DET = 137.98279634

1	4	-.8450	.1070	.2275
---	---	--------	-------	-------

INVERSE  
 $K_1 = 1$   
 $K_2 = 1$   
DET = 35.289000046

2	3	2.964	.457E-01	-.1475
---	---	-------	----------	--------

INVERSE  
 $K_1 = 1$   
 $K_2 = 1$   
DET = 42.328042367

2	4	-.6936	.3960E-02	.3057
---	---	--------	-----------	-------

INVERSE  
 $K_1 = 1$   
 $K_2 = 1$   
DET = 76.500227722

3	4	2.774	-.1938E-01	.1001
---	---	-------	------------	-------

TABLE 11. TRADEOFF FOR PAIR 1, 2

DATE 615 ID TC1C--IT  
TRADEOFF FOR  
PAIR 1 2

COEFF

	•36483	•06198	•08459			SCORE	0	1	2	3	4
	NO	ID TY		CL							
1	1	1 1	0 0	0 1		•8891		1			
2	2	2 1	0 0	0 1		•9285		2			
3	3	3 1	0 0	0 1		1•0131		3			
4	5	5 1	0 0	0 1		1•0357		4			
5	4	4 1	0 0	0 1		1•0877		5			
6	16	16 1	0 0	0 1		1•2610		6			
7	7	7 2	0 0	0 2		1•2669			①		
8	11	11 1	0 0	0 1		1•2836		7			
9	6	6 1	0 0	0 1		1•3741		8			
10	10	10 1	0 0	0 1		1•3908		9			
11	21	21 3	0 0	0 3		1•4696				1	
12	17	17 1	0 0	0 1		1•4922		10			
13	30	30 3	0 0	0 3		1•5089				2	
14	12	12 2	0 0	0 2		1•5374					
15	9	9 2	0 0	0 2		1•5826					
16	13	13 2	0 0	0 2		1•6220					
17	34	34 3	0 0	0 3		1•6555				3	
18	18	18 2	0 0	0 2		1•6614					
19	26	26 2	0 0	0 2		1•7007				5	
20	8	8 1	0 0	0 1		1•7744				6	
21	38	38 3	0 0	0 3		1•7795					
22	27	27 1	0 0	0 1		1•7853					
23	14	14 2	0 0	0 2		1•7912					
24	37	37 3	0 0	0 3		1•8021				4	
25	22	22 2	0 0	0 2		1•8079				5	
26	31	31 2	0 0	0 2		1•8473					
27	39	39 3	0 0	0 3		1•8641					
28	23	23 2	0 0	0 2		1•8925				6	
29	19	19 2	0 0	0 2		1•9151				10	
30	35	35 2	0 0	0 2		1•9939				11	
31	32	32 2	0 0	0 2		2•0165				12	
32	28	28 2	0 0	0 2		2•0391				13	
33	20	20 4	0 0	0 4		2•0843				14	
34	15	15 4	0 0	0 4		2•1295					1
35	24	24 4	0 0	0 4		2•2309					2
36	33	33 4	0 0	0 4		2•2703					3
37	29	29 4	0 0	0 4		2•2929					4
38	25	25 4	0 0	0 4		2•3155					5
39	36	36 4	0 0	0 4		2•5860					6

$$\text{const (4,5)} = \frac{1.4922 + 1.5374}{2} = 1.51$$

TABLE 12. TRADEOFF FOR PAIR 1, 2

DATE 615 ID TC1C--11  
TRADEOFF FOR  
PAIR 1 3

COEFF

.31100		.17789		-.04246		CL	SCORE	0	1	2	3	4
NO	ID	TY										
1	6	6	1	0	0	0	.6404		1			
2	8	8	1	0	0	0	.6485		2			
3	1	1	1	0	0	0	.6748		3			
4	5	5	1	0	0	0	.8173		4			
5	7	7	2	0	0	0	.9732			1		
6	9	9	2	0	0	0	.9537			2		
7	4	4	1	0	0	0	.9881		5			
8	3	3	1	0	0	0	1.0306		6			
9	2	2	1	0	0	0	1.0731		7			
10	15	15	4	0	0	0	1.0972					1
11	10	10	1	0	0	0	1.2590		8			
12	14	14	2	0	0	0	1.2670			3		
13	13	13	2	0	0	0	1.3520			4		
14	12	12	2	0	0	0	1.3944			5		
15	11	11	1	0	0	0	1.5218		9			
16	20	20	4	0	0	0	1.5379					2
17	19	19	2	0	0	0	1.6228			6		
18	25	25	4	0	0	0	1.6309					3
19	24	24	4	0	0	0	1.6733					4
20	16	16	1	0	0	0	1.7422	3				
21	18	18	2	0	0	0	1.7502		10			
22	17	17	1	0	0	0	1.8351	2		7		
23	23	23	2	0	0	0	1.8432	1				
24	29	29	4	0	0	0	1.8512					5
25	22	22	2	0	0	0	1.8856			9		
26	28	28	2	0	0	0	1.9786	1		10		
27	21	21	3	0	0	0	2.0555				1	
28	33	33	4	0	0	0	2.0716					6
29	27	27	1	0	0	0	2.1060	1				
30	36	36	4	0	0	0	2.1221					
31	26	26	2	0	0	0	2.1484					
32	32	32	2	0	0	0	2.1989					
33	31	31	2	0	0	0	2.2839					
34	35	35	2	0	0	0	2.4193					
35	30	30	3	0	0	0	2.4537	1				
36	34	34	3	0	0	0	2.5891	2				2
37	37	37	3	0	0	0	2.7246	3				3
38	39	39	3	0	0	0	2.9024					4
39	38	38	3	0	0	0	2.9449					5

$$\text{const (4,6)} = \frac{1.8351 + 2.4537}{2} = 2.14$$



TABLE 13. TRADEOFF FOR PAIR 1, 4

DATE 615 ID TC10--TT  
TRADEOFF FOR  
PAIR 1 4

LBEFF		.10700		.22754		CL	SCORE	0 1 2 3 4				
--84505	N9	ID	TY									
1	2	2	1	0	0	0	1	.3726	1			
2	1	1	1	0	0	0	1	.3861	2			
3	3	3	1	0	0	0	1	.6001	3			
4	5	5	1	0	0	0	1	.7206	4			
5	4	4	1	0	0	0	1	.8276	5			
6	16	16	1	0	0	0	1	1.0281	6			
7	11	11	1	0	0	0	1	1.1486	7			
8	7	7	2	0	0	0	2	1.2827		1		
9	30	30	3	0	0	0	3	1.4561			1	
10	21	21	3	0	0	0	3	1.4696			2	
11	10	10	1	0	0	0	1	1.4967	3			
12	17	17	1	0	0	0	1	1.5902	9			
13	6	6	1	0	0	0	1	1.6308	10			
14	34	34	3	0	0	0	3	1.7906			3	
15	12	12	2	0	0	0	2	1.8312		2		
16	38	38	3	0	0	0	3	2.0046			4	
17	26	26	2	0	0	0	2	2.0317		3		
18	18	18	2	0	0	0	2	2.0453		4		
19	13	13	2	0	0	0	2	2.0588		5		
20	9	9	2	0	0	0	2	2.0723		6		
21	37	37	3	0	0	0	3	2.1252			5	
22	39	39	3	0	0	0	3	2.2322			6	
23	27	27	1	0	0	0	1	2.2593	11			
24	31	31	2	0	0	0	2	2.3663		7		
25	22	22	2	0	0	0	2	2.3798		8		
26	14	14	2	0	0	0	2	2.5139		9		
27	23	23	2	0	0	0	2	2.6073		10		
28	8	8	1	0	0	0	1	2.6479	12			
29	35	35	2	0	0	0	2	2.7008		11		
30	19	19	2	0	0	0	2	2.7279		12		
31	32	32	2	0	0	0	2	2.8213		13		
32	28	28	2	0	0	0	2	2.9419		14		
33	20	20	4	0	0	0	4	3.1829			1	
34	15	15	4	0	0	0	4	3.4240			2	
35	33	33	4	0	0	0	4	3.5039			3	
36	24	24	4	0	0	0	4	3.5175			4	
37	29	29	4	0	0	0	4	3.6245			5	
38	25	25	4	0	0	0	4	3.7450			6	
39	36	36	4	0	0	0	4	4.2935			7	

0 — 2.6479 — 12

0 — 2.9419 — 14

2.6479 — 12

2.9419 — 14

const (4,7) =  $\frac{2.6479 + 3.1829}{2}$

= 2.92

TABLE 14. TRADEOFF FOR PAIR 2, 3

DATE 615 ID TC1C--TT  
TRADEOFF FOR  
PAIR 2 3

COEFF

2.96373		.04507		-.14755		CL	SCORE	C	1	2	3	4
NB	ID	TY										
1	15	15	4	0	0	0	1.1111					1
2	36	36	4	0	0	0	1.2339					2
3	25	25	4	0	0	0	1.2463					3
4	8	8	1	0	0	0	1.2710		1			
5	24	24	4	0	0	0	1.3938					4
6	29	29	4	0	0	0	1.4389					5
7	20	20	4	0	0	0	1.4963					6
8	33	33	4	0	0	0	1.6315					7
9	14	14	2	0	0	0	1.7013			1		
10	9	9	2	0	0	0	1.7587			2		
11	19	19	2	0	0	0	1.7914			3		
12	6	6	1	0	0	0	1.8161		2			
13	28	28	2	0	0	0	1.8815			4		
14	23	23	2	0	0	0	1.9840			5		
15	13	13	2	0	0	0	1.9964			6		
16	32	32	2	0	0	0	2.0741			7		
17	22	22	2	0	0	0	2.1316			8		
18	12	12	2	0	0	0	2.1439			9		
19	7	7	2	0	0	0	2.1562			10		
20	18	18	2	0	0	0	2.2340			11		
21	10	10	1	0	0	0	2.2464		3			
22	35	35	2	0	0	0	2.2668			12		
23	27	27	1	0	0	0	2.3242		4			
24	31	31	2	0	0	0	2.3692			13		
25	5	5	1	0	0	0	2.4063		5			
26	4	4	1	0	0	0	2.4513		6			
27	26	26	2	0	0	0	2.4717			14		
28	1	1	1	0	0	0	2.5087	0		7		
29	17	17	1	0	0	0	2.5291			8		
30	11	11	1	0	0	0	2.5865			9		
31	3	3	1	0	0	0	2.5989	0		10		
32	21	21	3	0	0	0	2.7217				1	
33	2	2	1	0	0	0	2.7464			11		
34	37	37	3	0	0	0	2.7545				2	
35	16	16	1	0	0	0	2.7792			12		
36	39	39	3	0	0	0	2.7995				3	
37	34	34	3	0	0	0	2.8569				4	
38	38	38	3	0	0	0	2.9471				5	
39	30	30	3	0	0	0	2.9594				6	

$$\text{const (5,6)} = \frac{2.4717 + 2.7217}{2} = 2.60$$

TABLE 15. TRADEOFF FOR PAIR 2, 4

DATE 615 ID TC10--TT  
TRADEOFF FOR  
PAIR 2 4

COEFF

-.69355 .00396 .30567

	NO	ID	TY			CL
1	2	2	1	0	0	1
2	1	1	1	0	0	1
3	3	3	1	0	0	1
4	16	16	1	0	0	1
5	30	30	3	0	0	3
6	5	5	1	0	0	1
7	4	4	1	0	0	1
8	11	11	1	0	0	1
9	21	21	3	0	0	3
10	34	34	3	0	0	3
11	38	38	3	0	0	3
12	17	17	1	0	0	1
13	37	37	3	0	0	3
14	39	39	3	0	0	3
15	7	7	2	0	0	2
16	10	10	1	0	0	1
17	26	26	2	0	0	2
18	12	12	2	0	0	2
19	18	18	2	0	0	2
20	27	27	1	0	0	1
21	31	31	2	0	0	2
22	6	6	1	0	0	1
23	13	13	2	0	0	2
24	22	22	2	0	0	2
25	35	35	2	0	0	2
26	9	9	2	0	0	2
27	23	23	2	0	0	2
28	32	32	2	0	0	2
29	14	14	2	0	0	2
30	19	19	2	0	0	2
31	28	28	2	0	0	2
32	8	8	1	0	0	1
33	20	20	4	0	0	4
34	33	33	4	0	0	4
35	24	24	4	0	0	4
36	29	29	4	0	0	4
37	15	15	4	0	0	4
38	25	25	4	0	0	4
39	36	36	4	0	0	4

SCORE 0 1 2 3 4

.2433	1			
.5410	2			
.5489	3			
.5448	4			
.5206			1	
.8507	5			
.8546	6			
.8665	7			
.8784			2	
.8902			3	
.8982			4	
1.1801	3			
1.1999			5	
1.2038			6	
1.4660		1		
1.4739	3			
1.4937		2		
1.7235		3		
1.7914		4		
1.7993	10			
1.8133		5		
2.0733	11			
2.0892		6		
2.1011		7		
2.1129		8		
2.3869		9		
2.4167		10		
2.4147		11		
2.7105		12		
2.7084		13		
2.7164		14		
3.3198	12			
3.3317			1	
3.6294			2	
3.6334			3	
3.9232			4	
3.9351			5	
4.2527			6	

$$\left. \begin{array}{l} 0 \quad 2.7164 \\ 0 \quad 3.3198 \end{array} \right\} \text{const (5,7)} = \frac{2.7164 + 3.3198}{2} = 3.02$$

TABLE 16. TRADEOFF FOR PAIR 3, 4

DATE 615 ID TC1C--1T  
TRADEOFF FOR  
PAIR 3 4

COEFF

2.77429	-.01938	.10000	CL	SCORE	0	1	2	3	4
1 30	30 3	0	0	2.9227				1	
2 38	38 3	0	0	2.9647				2	
3 2	2 1	0	0	2.9777		1			
4 16	16 1	0	0	3.0003		2			
5 34	34 3	0	0	3.0035				3	
6 21	21 3	0	0	3.0416				4	
7 39	39 3	0	0	3.0648				5	
8 3	3 1	0	0	3.0778		3			
9 37	37 3	0	0	3.0842				6	
10 1	1 1	0	0	3.1145	0	4			
11 11	11 1	0	0	3.1197		5			
12 4	4 1	0	0	3.1779		6			
13 17	17 1	0	0	3.1811		7			
14 5	5 1	0	0	3.1972		8			
15 26	26 2	0	0	3.2424			1		
16 31	31 2	0	0	3.3231			2		
17 10	10 1	0	0	3.3393		9			
18 27	27 1	0	0	3.3425		10			
19 7	7 2	0	0	3.3781			3		
20 18	18 2	0	0	3.3813			4		
21 35	35 2	0	0	3.4038			5		
22 12	12 2	0	0	3.4200			6		
23 22	22 2	0	0	3.4620			7		
24 13	13 2	0	0	3.5201			8		
25 32	32 2	0	0	3.5233			9		
26 23	23 2	0	0	3.5621		10			
27 6	6 1	0	0	3.5976		11			
28 28	28 2	0	0	3.6428			11		
29 9	9 2	0	0	3.6590			12		
30 19	19 2	0	0	3.6815			13		
31 14	14 2	0	0	3.7203			14		
32 33	33 4	0	0	3.8236	0				1
33 20	20 4	0	0	3.8817					2
34 29	29 4	0	0	3.9431					3
35 24	24 4	0	0	3.9624					4
36 8	8 1	0	0	3.9786		12			
37 25	25 4	0	0	4.0625					5
38 36	36 4	0	0	4.1045					6
39 15	15 4	0	0	4.1207					7

$$\text{const (6,7)} = \frac{3.0842 + 3.8236}{2} = 3.45$$

From Equations (27) and (28), the values of  $W_{0ij}^{CTC}$  are computed, using  $\tilde{W}_{0ij}^{CTC}$  and  $\text{const}(i, j)$ , as

$$W_{0ij}^{CTC} = \tilde{W}_{0ij}^{CTC} + \text{const}(i, j) .$$

Thus,

$(i, j)$	$-\tilde{W}_{0ij}^{CTC}$	$\text{const}(i, j)$	$W_{0ij}^{CTC}$
1, 2	0.3648	1.51	1.145
1, 3	0.311	2.144	1.833
1, 4	-0.845	2.92	3.765
2, 3	2.964	2.60	-0.364
2, 4	-0.6936	3.02	3.714
3, 4	2.774	3.45	0.676

The coefficients are normalized as discussed in Appendix A to get direction numbers and distance from origin for each discriminant:

$(i, j)$	$  \tilde{W}_{ij}^{CTC}  _I$	$(W_{1ij}^{CTC})^n$	$(W_{2ij}^{CTC})^n$	$(W_{0ij}^{CTC})^n$
(1, 2)	0.105	0.59	0.81	10.9
(1, 3)	0.183	0.97	-0.23	10.02
(1, 4)	0.251	0.43	0.91	15.0
(2, 3)	0.154	0.29	-0.96	- 2.364
(2, 4)	0.306	-0.013	0.999	12.14
(3, 4)	0.102	-0.19	0.981	6.63



where

$$\|\vec{w}_{ij}^{CTC}\|_I = \left[ (w_{1ij}^{CTC})^2 + (w_{2ij}^{CTC})^2 \right]^{1/2}$$

$$(w_{kij}^{CTC})^n = w_{kij}^{CTC} / \|\vec{w}_{ij}^{CTC}\|_I, \quad k = 0, 1, 2$$

and

$$d(0, W) = |(w_{0ij}^{CTC})^n|.$$

Using the foregoing table, the linear discriminants are plotted in Figures 9-a and 9-b.

The piecewise linear discriminants are those separating classes 1, 2 from old class  $C_3$  ( $C_3$  and  $C_4$  comprise old class  $C_3$ ); they are shown as heavy dashed lines in Figure 8. Note, as discussed in Appendix B, the discriminant for class-pair  $C_3$  and  $C_4$  is not needed (only the original three classes are of interest).

The linear discriminants for the extended problem are plotted in Figure 9-b in solid straight lines. Note that the discriminant for sub-class pair (1, 3) was chosen using the maximum interval to compute  $\text{const}(1, 3)$ . This choice arose from the uncertainty as to the representativeness of the type 6 data, since so few points comprise the class. The data point #27 is far removed from the rest of the class and, since the data for type 4 are representative, can be considered as an anomalous point. Thus, points #21 and #27 were ignored in choosing  $\text{const}(1, 3)$ . The highly localized density of the rest of type 6 suggests representativeness, even though the  $\frac{M3}{M} = \frac{6}{39}$  ratio is small; thus these points were used to characterize type 6.

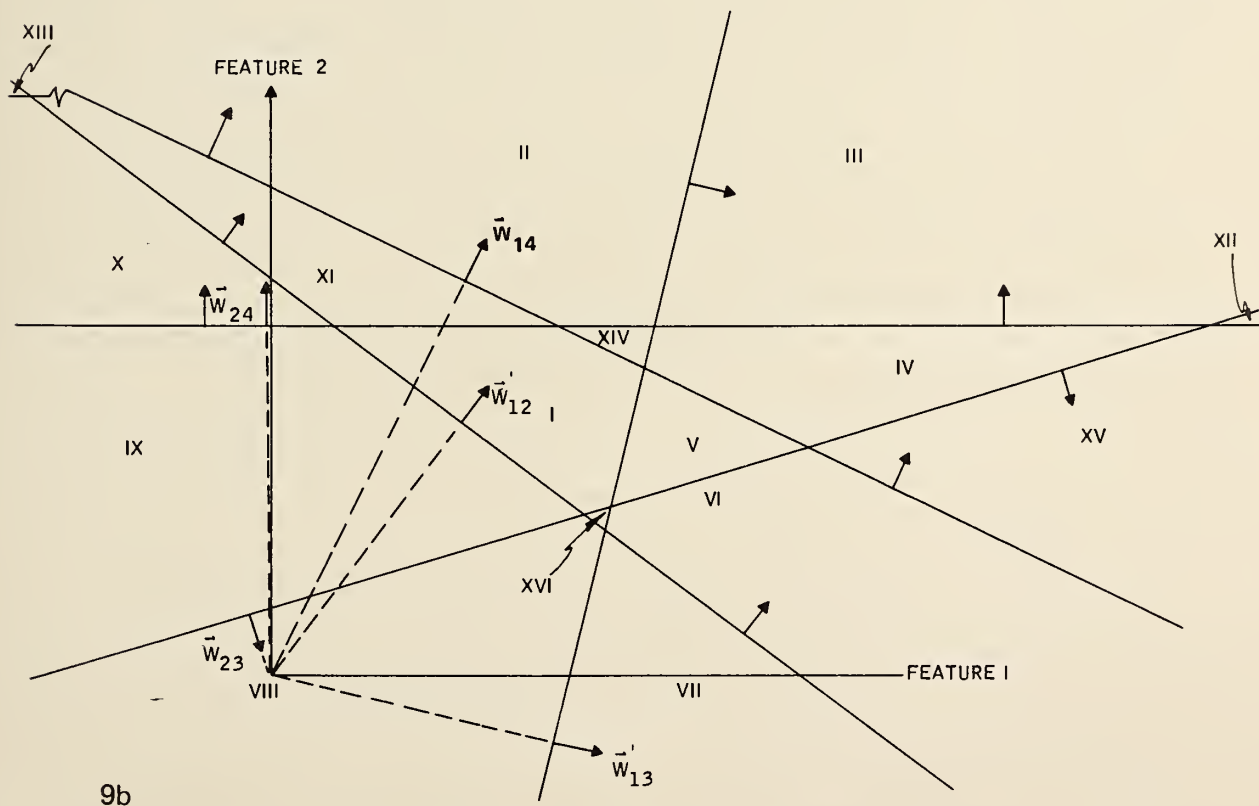
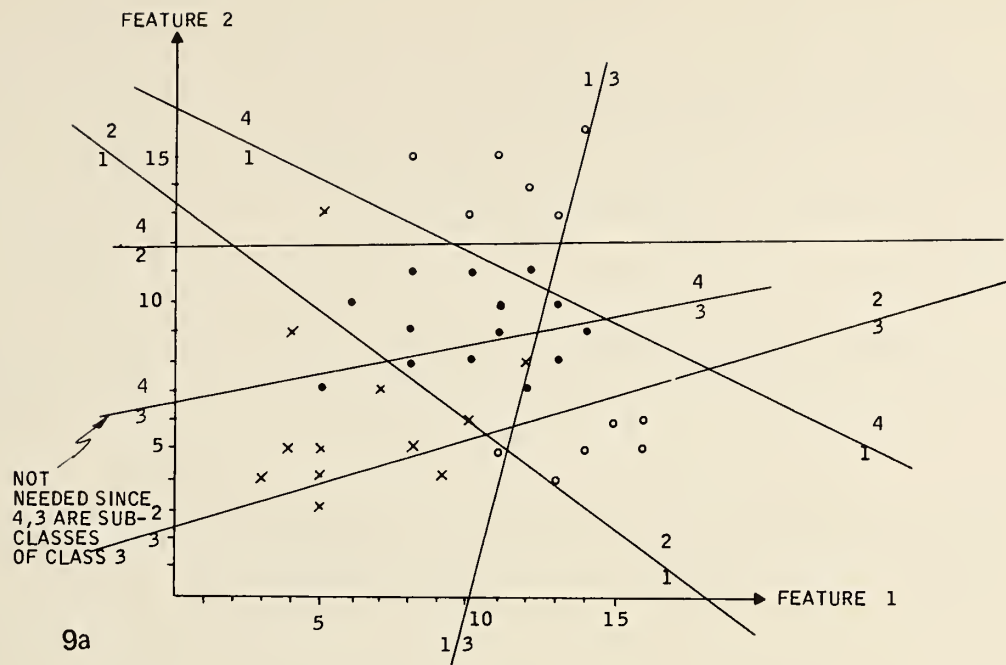


Figure 9. Linear Discriminants for Extended Example

### 3.3.3 The Layered Machine Classifier

Now that the piecewise linear discriminants

$$\Lambda_{ij}(X) = -W_{0,i,j} + W_{1,i,j} X_1 + W_{2,i,j} X_2 \begin{matrix} C_j \\ > \\ < \\ C_i \end{matrix} 0$$

have been determined with  $(i, j) = (1, 2), (1, 3), (1, 4), (2, 3), (2, 4)$ , for the three-class problem, we are interested in using them to classify new data points whose class membership is not known ahead of time. From the expression for  $\Lambda_{ij}(X)$ , we know that  $C_j$  lies on the positive side of the linear discriminant and  $C_i$  on the negative side (see Figure 9b). Further, each  $\Lambda_{ij}(X)$ , for a given  $X$ , allows us to remove a class from consideration; i. e.,  $e' = \{C'_1, C'_2, C'_3, C'_4\}$  and remembering that  $C_1 = C'_1, C_2 = C'_2, C_3 = C'_3 \vee C'_4$ ,

$$\Lambda_{ij}(X) > 0 \Rightarrow X \in \text{one of } e' - \{C'_i\}$$

$$\Lambda_{ij}(X) < 0 \Rightarrow X \in \text{one of } e' - \{C'_j\} .$$

Define the Boolean variable

$$B_{ij}(X) = \begin{cases} 1 ; \Lambda_{ij}(X) > 0 \\ 0 ; \Lambda_{ij}(X) < 0 \end{cases}$$

for each  $\Lambda_{ij}$ ,  $(i, j) = (1, 2), (1, 3), (1, 4), (2, 3)$ , and  $(2, 4)$ .

Then

$$B_{ij}(X) = 1 \Rightarrow X \in e' - \{C_i'\}$$

$$B_{ij}(X) = 0 \Rightarrow X \in e' - \{C_j'\}$$

and we can use the  $B_{ij}(X)$  to determine the class membership of  $X$ . This is done by using the truth table for the  $B_{ij}(X)$ : ( $e = \{C_1, C_2, C_3\}$ )

$(i, j)$	$B_{ij} = 1$	$B_{ij} = 0$
$(1, 2)$	$e - \{C_1\} = \{C_2, C_3\}$	$e - \{C_2\} = \{C_1, C_3\}$
$(1, 3)$	$\{C_2, C_3\}$	$\{C_1, C_2\}$
$(1, 4)$	$\{C_2, C_3\}$	$\{C_1, C_3\}$
$(2, 3)$	$\{C_1, C_3\}$	$\{C_1, C_2\}$
$(2, 4)$	$\{C_1, C_3\}$	$\{C_1, C_2\}$

From the truth table, if  $B_{ij} = 0$  for each  $(i, j)$  pair, we conclude

$$X \in \{C_1, C_3\} \text{ and } \{C_1, C_2\}$$

or

$$X \in \{C_1, C_3\} \wedge \{C_1, C_2\} = \{C_1\} .$$

Thus, we must conclude  $X \in C_1$ .

Similarly, if

$$B_{1,j} = 0, \quad j = 2, 3, 4$$

and

$$B_{2,j} = 1, \quad j = 3, 4$$

$$X \in \{C_1, C_3\} \wedge \{C_1, C_2\} = \{C_1\}.$$

Thus, we can define the Boolean function  $F_{C_1}(X)$  such that

$$F_{C_1}(X) = \begin{cases} 1, & X \in C_1 \\ 0, & X \notin C_1 \end{cases}$$

as

$$F_{C_1}(X) = \overline{B_{1,2}(X)} \wedge \overline{B_{1,3}(X)} \wedge \overline{B_{1,1}(X)} \wedge (A \vee B)$$

$$A = (\overline{B_{2,3}} \wedge \overline{B_{2,4}}) \vee (\overline{B_{23}} \wedge B_{24})$$

$$B = (\overline{B_{23}} \wedge B_{24}) \vee (\overline{B_{23}} \wedge \overline{B_{24}}),$$

hence,  $A \vee B = 1$

and

$$F'_{C_1}(X) = \overline{B_{1,2}(X)} \wedge \overline{B_{1,3}(X)} \wedge \overline{B_{14}(X)} (= F_{C_1}(X)).$$

By analyzing the truth table in a similar manner as for  $F_{C_1}(X)$ , we determine

$$F'_{C_2}(X) = B_{12}(X) \wedge \overline{B_{23}(X)} \wedge \overline{B_{24}(X)} (= F_{C_2}(X))$$



$$\begin{aligned}
F_{C_3}(X) &= F'_{C_3} \vee F'_{C_4}(X) \\
&= (B_{13}(X) \wedge B_{23}(X)) \vee (B_{14}(X) \wedge B_{24}(X)) .
\end{aligned}$$

Using the  $F_{C_i}(X)$ , we can classify  $X$  by the following table.

$F_{C_1}$	$F_{C_2}$	$F_{C_3}$	X in class
1	0	0	$C_1$
0	1	0	$C_2$
0	0	1	$C_3$

Note that with reference to Figure 9b,

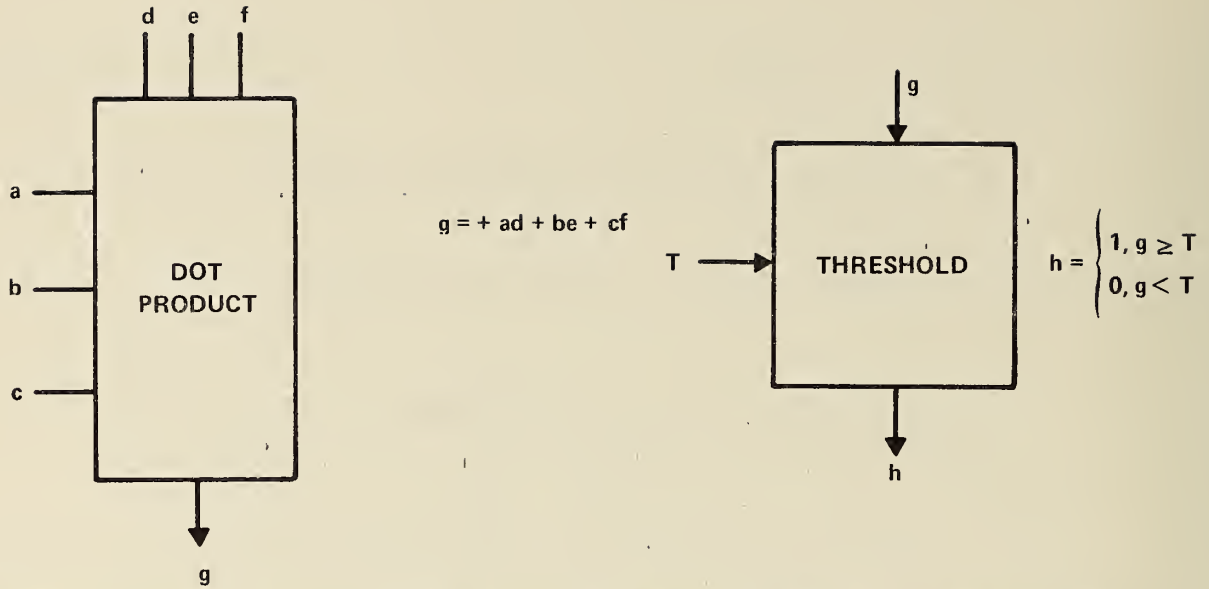
$$F_{C_1}(X) = 1 \Rightarrow X \text{ falls in one of regions X, IX, VIII}$$

$$F_{C_2}(X) = 1 \Rightarrow X \text{ falls in one of regions I, XIV, V, IV}$$

$$F_{C_3}(X) = 1 \Rightarrow X \text{ falls in one of regions II, III, XV, VI, XII, XIII,}$$

hence the  $F_{C_i}$  define disjoint regions of the space, and if one of the  $F_{C_i}$  is true (=1), the others must be false (=0). Thus,  $F_{C_3}(X) = \overline{F_{C_1}(X) \vee F_{C_2}(X)}$ .

Using this fact, a schematic for the classifier is given in Figure 10, computing  $F_{C_3}(X)$  using  $F_{C_1}(X)$  and  $F_{C_2}(X)$ . In the figure,



The classifier in Figure 10 is termed a layered machine classifier, since the computation is carried out in successive stages or layers:

- layer 1 :  $\Lambda_{ij}(X)$  computation
- layer 2 : threshold decision
- layer 3 : Boolean function computation
- layer 4 : encoded classification of  $X$ .

Figure 11 shows a flow chart for performing the same operations in software.

In general, the design of the layered machine classifier can be carried out just as done for our simple example;

$$F'_{C_i}(X) = \left( \bigwedge_{\substack{j=i+1 \\ (i,j) \notin L}}^N \overline{B_{ij}(X)} \right) \wedge \left( \bigwedge_{\substack{j=1 \\ (j,i) \notin L}}^{i-1} B_{ji}(X) \right)$$

with each

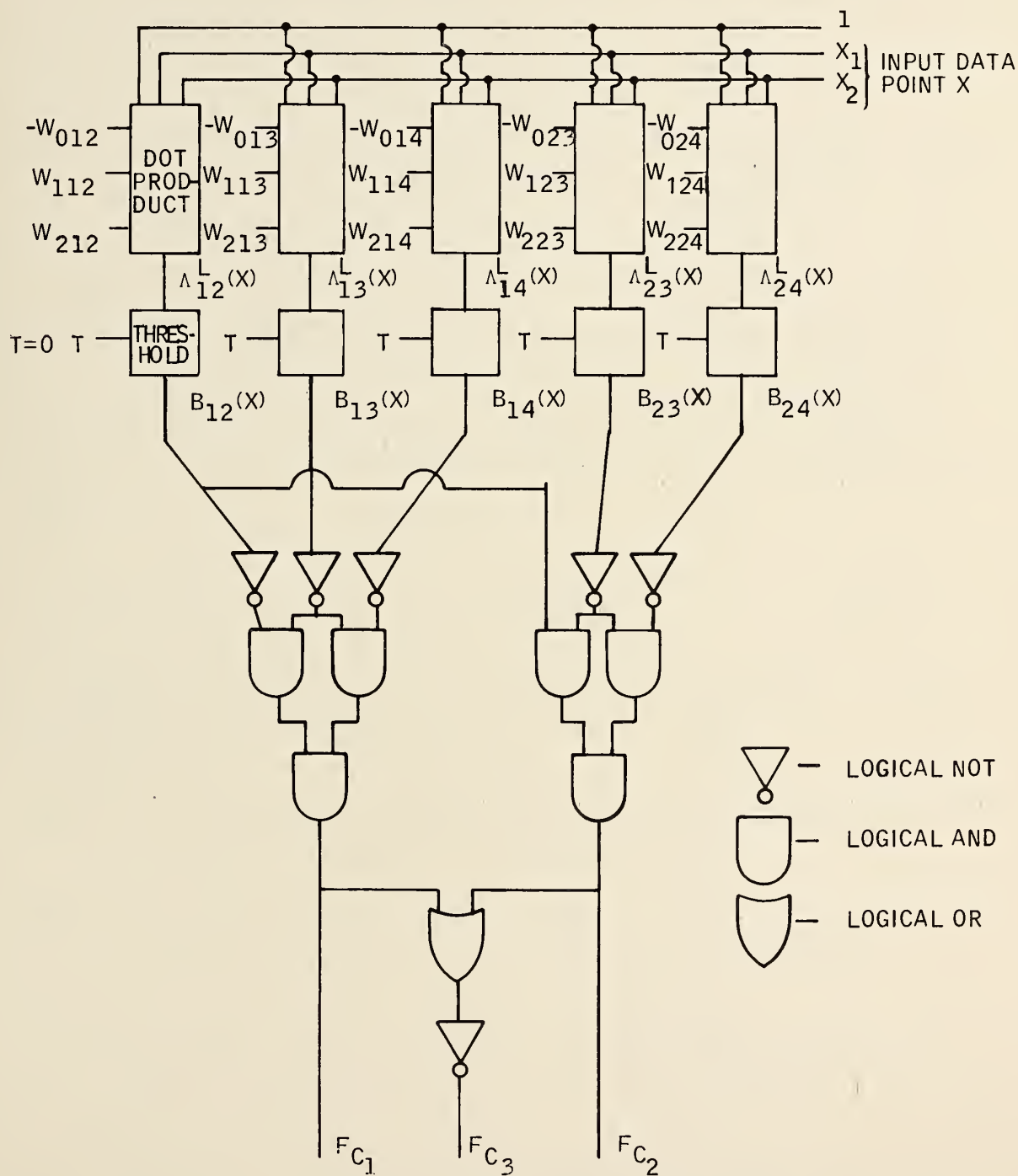


Figure 10. Hardware Logic Implementation of Piecewise Linear Classifier

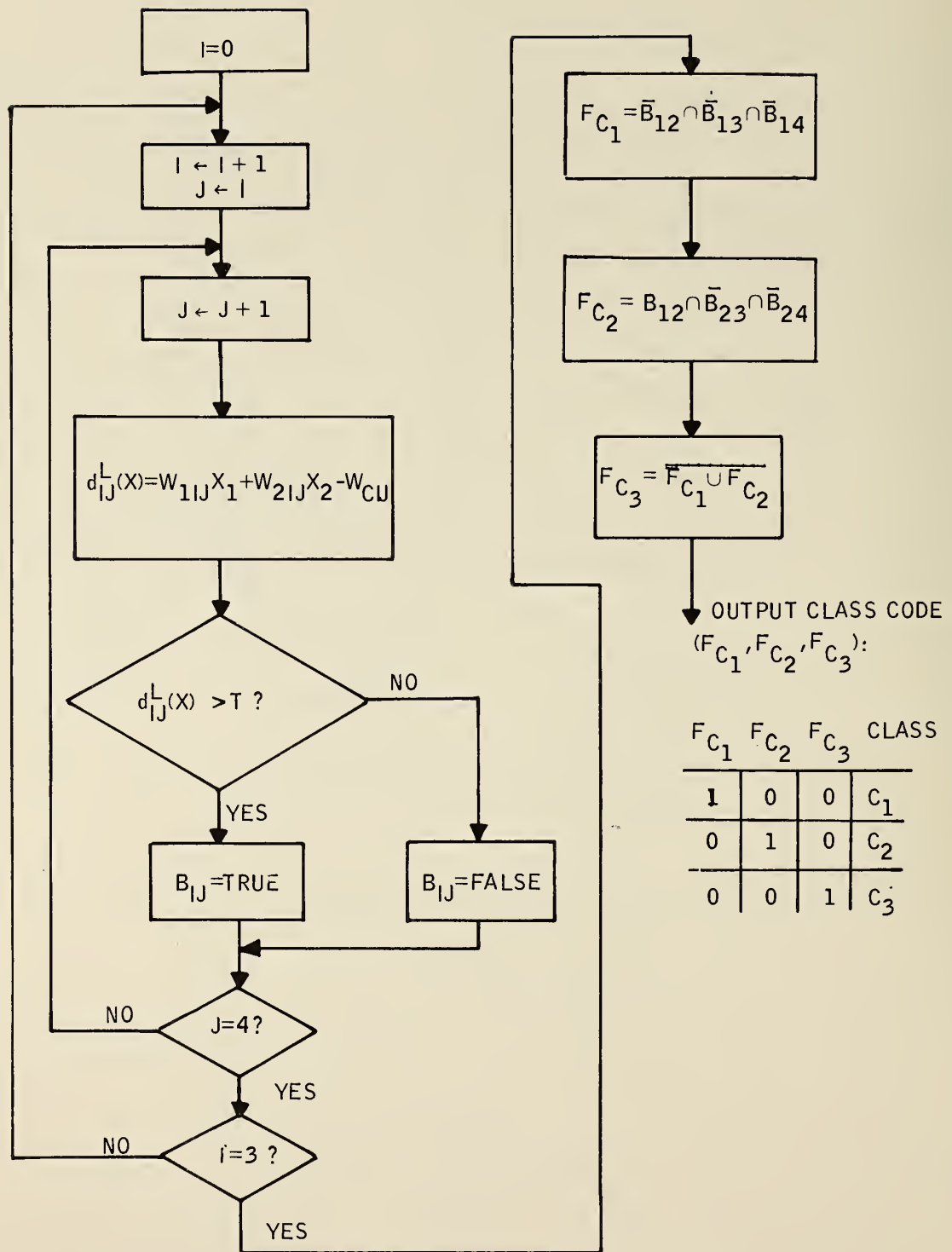


Figure 11, Flow Chart Corresponding to Figure 10

$$B_{ij}(X) = \begin{cases} 1 & ; \quad \Lambda_{ij}(X) > 0 \\ 0 & ; \quad \Lambda_{ij}(X) < 0 \end{cases}$$

where for  $(i, j)$  pairs such that  $\Lambda_{ij}(X)$  is not needed (as is the case for  $\Lambda_{34}(X)$  in our example problem) we define  $L = \{(i, j)\}$ .

One defect of this approach to the  $F'_{C_j}(X)$  is that the resulting Boolean functions are not necessarily minimal. Thus, the CTC program includes a logical mode whose output can be used to determine minimal expressions  $F^M_{C_j}(X)$  using the  $F'_{C_j}(X)$  as a guide. This minimization procedure is discussed in Chapter 4, CTC Operation, and is based on the Karnaugh mapping technique.

### 3.4 INTERACTIVE DESIGN OF CLASSIFIERS

The classifier design example of the previous section was carried out using the following sequence of steps:

- 1) Analyze the labeled samples and the results from previous attempts at classification that were not satisfactory.
- 2) (i) Based on the analysis in (1), define new subclasses, consolidate old ones, etc. and design a new classifier, or  
  
(ii) Decide, on the basis of dimensionality tests and previous classification attempts to try different features or to collect more labeled samples (or both).
- 3) Iterate on (1) and (2) until a satisfactory classifier is designed or until enough failures warrant a redesign of the measurement system (for example, one might have to try a number



of sensors for vehicle detection before the successful inductive loop/phase modulator sensing system is discovered).

The foregoing design method requires intelligent (human) interaction with the classifier design program. The interaction is in the form of analyzing program results and making the necessary program control decisions. For the CTC program, input cards control the program (see Chapter 4), while line printer output provides the data for analyzing (i. e. , property listings of classifier results, projections, statistics) dimensionality, complexity, and error rates.

The overall process of iteratively studying the data and trying many classifier designs provides a vehicle for familiarizing the designer with a specific data set in a specific feature space. This familiarization process provides the basis for making control decisions such as 2-ii. Note that the resulting classifier is not necessarily one deemed optimal by results of statistical theory (see Appendices A, B). Its merit is that a large degree of human knowledge has, by the interactive process, been incorporated in the final design.

Since the major cost of solving a pattern recognition problem is usually for collecting the labeled samples<sup>\*</sup>, the cost of computer time required by an interactive design process is a small consideration. Further, and more to the point, in many pattern recognition problems, the human is the standard (recall the discussion of using 'humps' in phase-versus-time plots as features, from Chapter 1). He should be put in charge of the classifier design and evaluation, and in control of automated aids for the design and evaluation process (i. e. , the CTC program).

---

<sup>\*</sup>For example, collecting data for various vehicles in the data set used to design the bus classification required: 1) drivers and vehicles, 2) the design and construction of a prototype sensor system (inductive loop-phase modulation detection), and 3) data recording equipment and a careful monitoring of experiments for labelling purposes.

Finally, the interactive design approach, while removing the need for laborious computations and data bookkeeping, gets the most information from an expensive data set. (Note how a relatively small number of samples was used to determine a satisfactory classifier in the previous section.)

### 3.5 THE CTC PHILOSOPHY

The CTC program was developed using the following guidelines:

- 1) Human standards and judgment are vital to the design of successful recognition systems.
- 2) One should not attempt to deify techniques that have worked for specific problems; future developments will have to be incorporated in any interactive system that is to lead a useful life. The system must be flexible to be useful for more than one problem.

Thus, the CTC program, at its inception, was designed:

- 1) For interactive use.
- 2) (i) So that a wide variety of data may be used. For the bus detector design, CTC was used first to separate moving from stopped vehicles and then again to design a classifier for moving vehicles.

In addition, the CTC has also been used in classifying image segments to develop the Honeywell Autoscreener.

(ii) So that the CTC program is modular in form, allowing an alteration of classifier techniques and an introduction of new data analysis subroutines without requiring drastic revision of the driving program and existing subroutines.

Finally, the hierarchical approach to problem solving provides the philosophy guiding the use of the CTC program.

The Hierarchical approach to problem solving is simply: for a difficult problem, attempt to 'break-up' the problem into sub-problems which are more easily solved, and then combine these individual solutions to solve the original problem.

The procedure is illustrated in Chapter 2. Rather than attempt a piecewise linear discriminant solution directly (such techniques are known-see Meisel), we solved a collection of linear discriminant sub-problems, and, using Boolean algebra, combine the results into a way giving a piecewise linear solution to the problem originally of interest. The name Cascaded Threshold Classifier is an allusion to the resulting classifier form (when solved hierarchically). From Figure 10 we see a cascade of decision layers: first the linear discriminant decisions, then the Boolean decision network operating on the linear discriminant decisions. The overall solution for the bus classifier is also hierarchic. We first classify moving vehicles versus non-moving vehicles, and then classify moving vehicles as "bus" or "non-bus". Each stage uses a CTC-structure classifier, such as shown in Figure 10.

## CHAPTER 4

### DATA CARDS AND CONTROL CARDS USED IN THE CTC PROGRAM

The Cascaded Threshold Classifier (CTC) program obtains its data and its direction from input cards. This chapter describes the data and control card formats, their functions and how they interact when used together. Two examples are given at the end of the chapter to show how typical problems can be solved using CTC.

In essence, the CTC program is a collection of data-processing and data listing subprograms. A control card deck is used to select which subprograms are used and to control their operation. For a single "run," all control cards are read before any processing operations take place. Thus, with one exception, the order of the cards in the control deck is not important.\* Each kind of control card selects one of the processing or listing options. The number punched in the first three columns of the card specifies the kind of control card it is; thus building a control card deck is like writing a computer program - each control card contains one instruction with the operation code being given by the number in the first three columns and the remaining columns telling how the operation is to be performed.

Data are also input on cards. There are seven different types of data cards, and, like the control cards, the different types are identified by the number appearing in the first three columns of the card. There are two categories of data that are input on data cards - parameter values and experimental data. Parameter cards contain coefficients, probability values, the number

---

\*The exception is the -99 card (Execute card), which is always the last card of a deck.



of samples per class, subclass combination rules and feature selection parameters. These parameters are used in performing the functions selected by the control cards. The experimental data cards contain the feature values and identifying information (such as data type, sample number, etc.).

Any card that is blank or has only zeros in the first three columns is ignored by the program, regardless of what appears in the remaining columns. This allows the user to include comments in the input deck.

Each deck of control and data cards ends with a card having -99 in the first three columns. When a -99 card is read by the program, the program executes the selected functions using the data that was input or specified by the deck. When all the functions have been finished, the program returns to the card reader and will read and execute another input deck if one is present. In executing successive decks, the program has options to (1) use data that have been transformed by previous decks, (2) restore the original data, (3) append new data to that already in the computer, (4) clear and load new data.

The next two subsections of this chapter describe the data cards and the control cards. The following section presents a tailored example of classifying a set of two-dimensional data. The input deck, containing the control cards and data cards, and the resulting computer output are shown. The final subsection is another example which was constructed by appending a third feature value to each of the feature vectors used in the first example. This example shows how additional features can improve the classification accuracy.

Table 17 summarizes the control and data card function. Table 18 describes the card format in detail and should be useful as a quick reference when using the CTC program.



## DATA CARD FORMATS AND CONTENTS

### Background

The first three character positions (first three columns) on any input card are taken together as a single data field, as one item of data. The content of this first field indicates the function of the card and designates the use to be made of other information on the card. Generally, a positive function code in this first field indicates a data card, while negative function codes are used for program control cards. Any blanks or spaces in this first field are read as zeroes by the program; thus care must be taken to ensure proper interpretation of function codes in this first field. Excepting "minus one" cards, column four is always blank. Field two of any card (except a "minus one" card) starts with the first non-blank character after column four and ends with the first comma or blank character which follows a digit or a decimal point after column four. Field three starts with the first non-blank character after field two and ends with the first comma or blank character which follows a digit or a decimal point after field four. Other fields are similarly delimited by comma characters or blanks; this is known as "widthless" field specification. Fields which contain no values, empty fields, can be punched with no columns in the field. Thus an empty field can be represented by commas in adjacent card columns. An empty field is read as a field containing the number zero. Comma characters and decimal point characters are not permitted in the first three card columns.

The illustration below shows three ways that the same data can be input using the "widthless" format specification.

```
001 38,3,5,,,16,5,8,10,2
001 38 3 5 0 0 0 16 5 8 10 2
1 38 3 5, , , 16, 5, 8, 10, 2
```

## "One" Card Format

A "one" card has a function code of one, which may be punched in the first field as 001 or bb1 or b01 or 0b1 where b signifies the character blank or space. A "one" card function code may not be punched as 01b or 1bb or b1b or 010 or 100 since these would be read by the program as "ten" or "one hundred." "One" cards are used to specify feature vectors to the program. The second field of the "one" card contains a unique number, chosen by the user, to identify the feature vector being specified. Field three contains a number less than 21 which identifies the "type" of the vector. A single type-number may be used on more than one feature vector. The fourth field should contain a number which equals the quantity\* of features in the feature vector. If this value is greater than 10, the program will use only the first 10 features in the vector. Field five is used for another arbitrary identifying number, although this number may be the same as the value in field two. Field six is empty (Empty fields should be punched as described previously). Field seven is used only if secondary feature transformation is desired, (see discussion of "minus six" card). Future versions of the program may make use of fields six and seven, but for now they are unused. Field eight should contain the first feature of the feature vector. Field nine contains the second feature, field ten the third, and so on, until the end of the card (of course, only the quantity of features indicated in field four will be used by the program, and in no case will more than 10 features be used). Feature values in any feature vector are integers. If a non-integer value (such as 5.878) is punched into a card field to indicate a feature value, the value will be truncated to an integer (5.878 is truncated to just 5). In certain cases, features are calculated by the program; these calculated values are also truncated.

---

\* In discussing the data and control cards, the term "number" will be used generically to denote "a numerical item" while the term "quantity" will denote "a particular numerical value." This avoids phrases like "a number that equals the number of feature values."

001 38,3,5,,,16,5,8,10,2

Example of a "1" card

### "Two" Card Format

Cards having function codes equal to 002 or b02 or bb2 (b equals blank) are used to define coefficient vectors. If 10 coefficient vectors have already been defined, the card is ignored. "Two" cards may be used to define new features as linear combinations of old features. The second field of the "two" card contains the number of the feature to be created or replaced. Field three is a scaling factor for the new feature. Field four is a constant term which is used in determining the new feature. Field five contains the coefficient of the old feature number one in the expression defining the new feature. Field six contains the coefficient of the old feature number two, field seven, the coefficient for old feature number three, and so on. The new feature is formed by multiplying each element in the feature vector by its corresponding coefficient (fields five, six, seven, etc., of the "two" card); then adding all the terms produced, plus the constant from field four of the "two" card; then multiplying the resulting value by the scaling factor (field three of the "two" card) and substituting the final result for the feature specified in field two of the "two" card. New elements of feature vectors may be created in this way, or old elements may be replaced by new values. The substitution does not occur until all new elements have been calculated. The procedure is applied to each feature vector, although some elements (or even most elements) of the feature vector remain unchanged. A "five" card (defined later) must be used before any "two" cards, to indicate which features are to be used in the procedure.

Fields four and up may contain non-integer values; for example, .06737, 6.737E-2 and +6.737E-02 appearing in fields four and up will each be read as the non-integer value 0.06737. Non-integers in fields one, two or three will be truncated.

002 1, 10000, -3.18813, .17126, .09952, .06737

002 ,1000,,-.275,-.630,1.324

Example of a "2" card format

### "Three" Card Format

A card with function code 003 in its first field is used to assign weighting values to the various classes of feature vectors. This weighting value is proportional to the a priori probability for the corresponding class. The class probabilities are calculated by dividing each weight by the sum of all the weights. Normally the weight for a class will be equal to the quantity of samples in that class. The second field is a number indicating the quantity of classes to be used, and is also equal to the quantity of weighting elements on the "three" card. Field three contains weight for class one, field four contains the weight for class two, field five the weight for class three, and so on, with a weight factor for each class.

003 4, 12, 14, 6, 7

Example of a "3" card

### "Four" Card Format

Assignment of feature vector "types" to "classes" is done by using a class assignment card, having 004 in its first field. The second field of a "four" card indicates the quantity of feature vector types to be assigned to classes,



and must be 20 or less. The third field contains the class number for vectors of type 1. The fourth field contains the class number of vectors of type 2, fifth field is class for vector type 3, etc. All class numbers must be values between zero and ten inclusive; specified class numbers greater than ten are set equal to ten by the program. Also, if a class number is greater than the number of classes as specified in field two of the "three" card, the class number from field two of the "three" card is used. Note that several types may be assigned to the same class.

This type-to-class conversion is used to combine a prior designated subclasses into a single class for the classifier design. In designing the bus classifier each different vehicle tested was given a different type number. This helped to keep track of the way different kinds of vehicles were classified, but to design the classifier, all the busses were assigned to one class and all the non-busses were assigned to one class.

004 4, 1, 2, 3, 3

Example of a "4" card

#### "Five" Card Format

A card with its first field equal to 005 is used for feature selection. Field two of a "five" card contains a value (less than or equal to ten) indicating the quantity of features to be selected. Remaining fields contain values indicating the number of the selected feature. For example, assume feature vectors containing nine features are available. If we wish to use the first, third, fifth, seventh, and ninth features of these vectors, the "five" card would look like this: 005 5, 1, 3, 5, 7, 9. The feature vector which is then available to be used by the program contains only the five specified features instead of its original nine.



005 3, 1, 2, 4

Example of a "5" card format

### Two-Card Input Format

Besides the "one" card, feature vectors may be specified to the program in another way that uses two cards instead of one. These cards have first fields of 011 and 012, respectively. The second field of the "eleven" card contains a vector identifying number, the third field indicates the "type" of feature vector, and fields four, five, and six contain identifying information. The "twelve" card contains 012 in its first field, an identifier in the second field, the identifier from field two of the corresponding "eleven" card in field three and elements (features) of the feature vector in succeeding fields. More than one "twelve" card may correspond to a single "eleven" card, but the "eleven" card must precede all corresponding "twelve" cards.

This two card input format is useful when repeated experiments, with the same controlled conditions, are used to gain information about the statistical variations in the feature measurement process. In this situation a single "eleven" card is prepared to identify the test conditions. Then for each separate experiment performed under these conditions, a separate "twelve" card is prepared containing the features produced by the experiment. Using the two card format thus reduces the amount of key punching that has to be done to load the data into the program.

011 5, 3, 20, 12

"2" - Card Input Format

012 25,5,17,10,24,31,2500,2000,3000

## CONTROL CARD FORMATS AND CONTENTS

The first three character positions (first three columns) on any input card are taken together as a single data field, as one item of data. The content of this first field indicates the function code for the card, designating the use to be made of the remaining information on the card. Card inputs are used to establish such things as listing options, classifier method, class assignments, class probabilities, feature vector values. Generally, a data input card has a positive value in its first field, while program control cards contain negative values in the first field. Except for the card used to indicate date and identification, card fields after the first are defined as being widthless, which means the fields are delimited by symbols punched into the card (such as commas) rather than by external specifications (in the program code itself) of allowable field size.

### "Minus One" Card Format

The exception to the widthless field rule, as noted above, is for the card used to indicate the date and identification of a particular execution of the program. This identifier card has a structured format which must be used in order to ensure proper recognition by the program. As on all cards, the first three character positions contain the code for the card function; in this case (identifier card) this code is -1. This may be punched into the card several different ways, such as: -01 or b-1 (where b indicates blank or space) or -b1. Note that if -1b is punched into this first field, the program will treat the blank as a zero and will read the field as -10. An error will occur, since -10 is not an allowable code.

The second field of this card occupies four card columns, numbers 4, 5, 6, and 7; and should contain the date, in a format such as 0215 or 1103 (meaning February 15 or November 3), respectively. The third field starts in column 8 and contains any numbers or letters desired as an identification. Only the first eight characters of this identifier are used; characters after column 15 are ignored.

-010615TC2B--TT

Minus "1" Card Format

### "Minus Two" Card Format

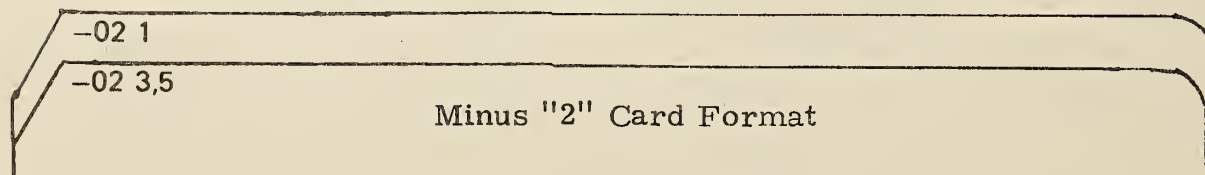
A card having a function code of -2 (punched in the first three columns as: -02 or b-2 or -b2 where b indicates blank or space) is used to indicate the mode of operation of the classifier. The second field of this "minus two" card starts in column 5 and contains the actual mode indication number. Values in field two and their meanings are:

- 1: Method 1 classifier: determines the coefficient vector defining the discriminant for each class pair using the minimum probability of error (least squares) decision rule (explained in chapter 3).
- 2: Method 2 classifier: determines the coefficient vector defining the discriminant for each class pair using the direction of smallest deviation decision rule.
- 3: Logical Mode Classifier: determines class membership by assigning a logical value (true or false) to each feature of a feature vector dependent upon the feature being greater than zero (true) or not (false). "True is represented by the digit 1 and "false" by the digit zero.
- 0: No classification is to be performed.

In the case of the logical mode classifier, another value is required on the "minus two" card: the quantity of features in each vector to be used in the logical mode. This value is punched as the third field on the card. For example, if the following "minus two" card is sensed by the program

-02 3,5

the logical mode will be run, using the first five features of each feature vector. For classifier methods 1 and 2, this third field is not required and if punched, is ignored by the program.



### "Minus Three" Card Format

A function code of -3 is used to control the output listing of the program. Values for the second field of a "minus three" card and meanings are:

- 5 List class dispersions
- 6 List class mean values and class covariances
- 7 List total data set covariance, list within class covariances and between-class covariances.

These values may occur in any order and should be separated by commas if more than one of them is punched. An example of a proper "minus three" card:

-03 5, 6, 7

specifies all options are to be performed. A negative value in the data field causes suppression of the corresponding option.

-03 5,6,7

Minus "3" Card Format

### "Minus Four" Card Format

A "minus four" card, with value -04 in its first field, is used to restore the set of feature vectors used in the previous execution. These vectors were saved when the previous execution was started; that is, when the previous "minus ninety-nine" card was encountered. A value equal to the quantity of vectors being saved was also stored at that time; when the vectors are restored, this value is used to control the quantity of vectors restored. Every time vectors are saved (at the beginning of an execution) values in the "save" area from a previous "save" are overwritten by the new values. If fewer vectors are saved in one instance than were saved the previous instance, errors could result during restoration if the count of saved vectors were not used. The restored data is placed in the feature vector array starting with the next available location in that array.



## Minus "4" Card Format

"Minus Five" Card Format

A "minus five" card is used to reset (to zero) the counter which indicates the next available location to write into the feature vector array. This counter is called NFV (number of feature vectors), and is incremented by one each time a feature vector is written into the feature vector array. The counter NFV starts equal to zero at the beginning of a run of several executions ("minus ninety-nine" cards) but is not reset (to zero) by the program encountering a "minus ninety-nine" card. A second counter, called Flag 1, is incremented by one each time a feature vector is read by the program from a data input card, and is reset to zero by a "minus ninety-nine" card. Thus Flag 1 has a value equal to the number of feature vector cards read by the program since the last execution. The NFV counter will be reset to zero (without using a "minus five" card) if no feature vector cards have been read (Flag 1 = 0) when a "minus four" (restore saved data) card is encountered. Thus restored data will always be written into the feature vector array starting at the beginning, unless one or more feature-vector cards have already been read by the program (Flag 1 = 0) for the current execution. A "minus five" card has the function code -5 punched in its first field and can be used before loading new data to be used for calculations.

## Minus "5" Card Format



### "Minus Six" Card Format

A function code of -6 or -06 is used to indicate whether or not a secondary transformation is to be performed on the feature data. The second field of a "minus six" card should contain a value of zero to indicate that no secondary transformation is desired, and should contain a value of one or two to indicate that a secondary feature transformation is to be performed on the input data feature vectors.

This transformation is determined by the value in the second field of the "minus six" card. The transformation is:

new feature one	= $1000 * \text{old feature three} / \text{FACT}$
new feature two	= $1000 * \text{old feature four} / \text{FACT}$
new feature three	= $1000 * \text{old feature six} / \text{old feature five}$
new feature four	= $1000 * \text{old feature seven} / \text{old feature five}$
new feature five	= $\text{old feature three} + \text{old feature four}$
new feature six	= $1000 * \text{old feature one} / \text{FACT}$
new feature seven	= $1000 * \text{old feature three} / \text{FACT} + 1000 * \text{old feature four} / \text{FACT}$

where FACT = old feature one + old feature two if the value in the second field of the "minus six" card is one and FACT = old feature two if the value in the second field is two. The transformation is performed on all feature vectors except for feature vectors read using "one" cards with the seventh field empty and feature vectors read using an "eleven" card with no corresponding "twelve" cards in the deck. Such feature vectors are suppressed by the transformation. During execution (following a "minus ninety-nine" card) this transformation is performed before any other operations. Omission of the "minus six" card from a deck is identical to the use of a "minus six" card with second field equal to zero. This transformation is used to convert the "raw" bus detector measurements into the normalized features used by the classifier.

-06 1

### Minus "6" Card Format

#### "Minus Seven" Card Format

A function code (first card field) of -7 designates a card which controls trade-off listing options for a given feature, the given feature being identified by the number in the second field of the "minus seven" card. The tradeoff listing produced by a "minus seven" card is a list of all feature vectors in increasing order based on the feature designated. For example, a "minus seven" card with its second field equal to 2 will cause the feature vectors to be listed with the vector having the smallest feature 2 value first and the vector with the largest feature 2 value last, and the other feature vectors ordered between these two. If the second field of a "minus seven" card contains a negative value, the selected feature is the negative of this value, and the listing gives the feature values.

-07 1

-07 -3

### Minus "7" Card Format

#### "Minus Eight" Card Format

A "minus eight" card may be used to project the feature vector set into a subspace of the space the vectors are in originally. Subspaces have fewer dimensions than the space containing them; for example, a plane (two dimensions) is a subspace of three-dimensional space; a line (one dimension) is a subspace of three-dimensional space; a line (one dimension) is a subspace of a plane, and is also a subspace of a three-dimensional space. Subspace projections are useful in observing data distributions; the data are constrained to lie in the subspace, thus removing a number of degrees of freedom equal to the difference in number of dimensions between the subspace and the total space. By also projecting the data into a complementary subspace, variations which

were lost in the subspace projection are shown. This allows a test to be made upon data which appear empirically to lie in a subspace of the total space, to determine if there is variation not contained in the subspace. For example, in three-dimensional space, an empirical data distribution may appear to lie in a single plane in the three space. Such a planar distribution would simplify the classifier, but, if the data in fact is not completely in the plane, variations which might be useful in the solution will be lost if the planar distribution is used alone. By using subspace projection into the supposed plane of the distribution, classification may be attempted with the planar distribution. By projecting to the complement of the planar subspace (in this case the complement of the planar subspace is a normal line to the plane) variations not in the plane may be observed, and may in fact provide an even better classification solution. This projection technique may be extended to as many dimensions as required; the general approach is to try a subspace solution and then see if the complement projection offers additional information. Of course, if the complement projection shows very little additional variation, the problem's solution can be simplified by projecting onto the subspace, thus reducing the degrees of freedom of the original distribution without loss of information.

The program uses a "minus eight" card to control projections; the second field of a "minus eight" card contains a value indicating that no projection is desired (second field = 0), or projection to a defined subspace (second field = 1) or projection to the orthogonal complement of a defined subspace (second field = 2). Omission of the "minus eight" card from a deck is identical to use of a "minus eight" with second field equal to zero.

Subspaces are defined by using coefficient cards ("two" cards) to define vectors which span the subspace. The fields of a "two" card used for projection are identical to those of a "two" card used to define a new feature (see "two" card discussion) except fields two and four (new feature number and added constant term) are ignored by the program when the card is used to define a



subspace spanning vector. Since all vectors pass through the origin, any two vectors will define (span) a two-dimensional plane; likewise any N vectors will define an N-dimensional subspace. By selecting subspaces within a multi-dimensional data space, projections may be performed to any subspace (or its orthogonal complement) within the data space, and printouts obtained showing the subspace data distribution.

-08 1

Minus "8" Card Format

#### "Minus Ninety-nine" Card Format

Function code -99 punched into the first field (first three columns) of a card signals that all data has been input to allow execution, and that execution should begin. No other fields are recognized or needed on a "minus ninety-nine" card.

-99

Minus "99" Card Format

#### TEST CASE 1

To better understand the steps performed by the CTC program, a test case was developed. This test case involved 39 samples (feature vectors), each having two features. This data set is easily visualized, and can be plotted on graph paper. Each of the 39 feature vectors was specified to the program by using: 1) an identifying number, 1-39; 2) an x value (feature 1); 3) a y value (feature 2); and 4) a type identifier. The type identifiers were arbitrarily chosen as 4, 5, 6 and each was assigned to a class: type 4 assigned to class 1, type 5 to class 2, type 6 to class 3.

TABLE 17. CONTROL AND DATA CARD FUNCTIONS

Code	Description or Use	Remains in Effect Until Explicitly Changed*	Action Immediate (Before Next Card is Processed)
-99	Execute	-	Yes
-8	Project (NPCON)	Yes	No
-7	Feature tradeoff	No	No
-6	Secondary feature (MODE 2)	No	No
-5	NFV = 0	No	Yes
-4	Restore feature data	No	Yes
-3	Listing options	Yes	No
-2	Method (KMODE)	Yes	No
-1	Identifier	Yes	Yes
1	Feature data	-	-
2	Coefficient data	-	-
3	Class card	Yes	Yes
4	Type card	Yes	Yes
5	Feature selection card	Yes	Yes
11	1st Feature card bus detector	-	Yes
12	2nd Feature card bus detector	-	Yes

\*That is to say, the option or control function or parameter is not changed or turned off by the program during, before or after execution.



TABLE 18. CONTROL CARD AND DATA CARD FORMAT SUMMARY

First Field	Second Field	Third Field	Fourth Field	Fifth Field	Sixth Field	Seventh Field	Eighth Field	Ninth Field	Tenth Field	Eleventh Field	Twelfth Field	Thirteenth Field
1	ID <sub>1</sub>	Type	Quantity of Features	ID <sub>2</sub>	Blank field	Blank field	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6
2	Feature number	Scale factor	Constant to be added	Coefficient 1	Coefficient 2	Coefficient 3	Coefficient 4	Coefficient 5	Coefficient 6	Coefficient 7	Coefficient 8	Coefficient 9
3	Quantity of classes	Quantity of vectors in class 1	Quantity of vectors in class 2	Quantity of vectors in class 3	Quantity of vectors in class 4	Quantity of vectors in class 5	Quantity of vectors in class 6	Quantity of vectors in class 7	Quantity of vectors in class 8	Quantity of vectors in class 9	Quantity of vectors in class 10	Blank to end of card
4	Quantity of types	Class NR for type 1	Class NR for type 2	Class NR for type 3	Class NR for type 4	Class NR for type 5	Class NR for type 6	Class NR for type 7	Class NR for type 8	Class NR for type 9	Class NR for type 10	Class NR for type 11
5	Quantity of features to be used	NR of 1st used feature	NR of 2nd used feature	NR of 3rd used feature	NR of 4th used feature	NR of 5th used feature	NR of 6th used feature	NR of 7th used feature	NR of 8th used feature	NR of 9th used feature	NR of 10th used feature	Blank to end of card
-1	Date	Identification	Code									
-2	Classifier method	Logical feature count										
-3	List Option	Option	Option									
-4												
-5	(Sets NFV = 0)											
-6	Primary or secondary features?											
-7	Feature NR for tradeoff listing											
-8	Subspace or complement											
11	ID <sub>1</sub>	Type	CHAR 1	CHAR 2	CHAR 3							
12	ID <sub>2</sub>	ID <sub>1</sub>	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7	Feature 8	Feature 9	Feature 10

Notes:

- "Two" card: A "five" card must occur before any "two" cards in the deck.
- "Two" card defines a new value for the feature number designated in field two or spanning get vectors. This new value is given by:
$$\text{new feature} = \text{scale factor} \left[ \text{constant} + (\text{coeff 1}) \cdot \text{feature 1} + (\text{coeff 2}) \cdot \text{feature 2} + (\text{coeff 3}) \cdot \text{feature 3} + (\text{coeff 4}) \cdot \text{feature 4} + \dots + (\text{coeff 10}) \cdot \text{feature 10} \right]$$
- "Minus three" card: negative values suppress options.
- "Minus four" card restores old data, which was saved during previous execution.
- "Minus six" card; second field = 0, primary; = 1 or = 2, secondary.
- "Minus eight" card; second field = 1, project to subspace  
second field = 2, project to complement of subspace.
- "One" card may contain up to ten features  
"Two" card may contain up to ten coefficients  
"Four" card may contain up to twenty types; but only a maximum of ten classes is allowed.  
"Minus nine" card may contain up to twenty types. Maximum number of features in a vector is ten.
- Fifth field of "one" card may be empty.

Data was input using 39 "one" cards, defined elsewhere. Class assignment was done by using a single "four" card, also defined elsewhere. Other cards required:

1. A "minus two" card, with mode = 1
2. A "minus three" card, specifying all options
3. A "minus six" card, specifying primary features
4. A "minus eight" card, specifying no projections
5. Four "minus seven" cards, one each for listings ordered by a) feature 1, b) feature 2; and one each for tradeoff listings for a) feature 1, b) feature 2.

-7	-1	listing ordered by feature 1
-7	-2	listing ordered by feature 2
-7	1	tradeoff listing for feature 1
-7	2	tradeoff listing for feature 2

6. A single "three" card, indicating the number of feature vectors of each type and the number of types.
7. A single "five" card, to indicate use of two features, namely feature 1 and feature 2, for calculations.
8. A single "minus one" card, with program date and ID.
9. A single "minus ninety-nine" card, following all other cards, to begin execution. Figure 12 shows this deck.





TABLE 20. TRADEOFF LISTING FOR FEATURE NUMBER -1

DATE 610 ID: TC1AM1													
TRADEOFF FOR FEATURE NUMBER -1													
	NO	CL	SCORE	1	2	3	4	5	6	7	8	9	10
1	1	1	3.0000	3	4	0	0	0	0	0	0	0	0
2	6	1	4.0000	4	9	0	0	0	0	0	0	0	0
3	5	1	4.0000	4	5	0	0	0	0	0	0	0	0
4	8	1	5.0000	5	13	0	0	0	0	0	0	0	0
5	7	2	5.0000	5	7	0	0	0	0	0	0	0	0
6	4	1	5.0000	5	5	0	0	0	0	0	0	0	0
7	3	1	5.0000	5	4	0	0	0	0	0	0	0	0
8	2	1	5.0000	5	3	0	0	0	0	0	0	0	0
9	9	2	6.0000	6	10	0	0	0	0	0	0	0	0
10	10	1	7.0000	7	7	0	0	0	0	0	0	0	0
11	15	3	8.0000	8	15	0	0	0	0	0	0	0	0
12	14	2	8.0000	8	11	0	0	0	0	0	0	0	0
13	13	2	8.0000	8	9	0	0	0	0	0	0	0	0
14	12	2	8.0000	8	8	0	0	0	0	0	0	0	0
15	11	1	8.0000	8	5	0	0	0	0	0	0	0	0
16	16	1	9.0000	9	4	0	0	0	0	0	0	0	0
17	20	3	10.0000	10	13	0	0	0	0	0	0	0	0
18	19	2	10.0000	10	11	0	0	0	0	0	0	0	0
19	18	2	10.0000	10	8	0	0	0	0	0	0	0	0
20	17	1	10.0000	10	6	0	0	0	0	0	0	0	0
21	25	3	11.0000	11	15	0	0	0	0	0	0	0	0
22	24	3	11.0000	11	14	0	0	0	0	0	0	0	0
23	23	2	11.0000	11	10	0	0	0	0	0	0	0	0
24	22	2	11.0000	11	9	0	0	0	0	0	0	0	0
25	21	3	11.0000	11	5	0	0	0	0	0	0	0	0
26	29	3	12.0000	12	14	0	0	0	0	0	0	0	0
27	28	2	12.0000	12	11	0	0	0	0	0	0	0	0
28	27	1	12.0000	12	8	0	0	0	0	0	0	0	0
29	26	2	12.0000	12	7	0	0	0	0	0	0	0	0
30	33	3	13.0000	13	13	0	0	0	0	0	0	0	0
31	32	2	13.0000	13	10	0	0	0	0	0	0	0	0
32	31	2	13.0000	13	8	0	0	0	0	0	0	0	0
33	30	3	13.0000	13	4	0	0	0	0	0	0	0	0
34	36	3	14.0000	14	16	0	0	0	0	0	0	0	0
35	35	2	14.0000	14	9	0	0	0	0	0	0	0	0
36	34	3	14.0000	14	5	0	0	0	0	0	0	0	0
37	37	3	15.0000	15	6	0	0	0	0	0	0	0	0
38	39	3	16.0000	16	6	0	0	0	0	0	0	0	0
39	38	3	16.0000	16	5	0	0	0	0	0	0	0	0



TABLE 21. TRADEOFF LISTING FOR FEATURE NUMBER -2

DATE	610	ID	TC1AM1
TRADEOFF	FOR	FEATURE	NUMBER
1	2	1	3.00000
2	30	3	4.00000
3	16	1	4.00000
4	3	1	4.00000
5	1	1	4.00000
6	38	3	5.00000
7	34	3	5.00000
8	21	3	5.00000
9	11	1	5.00000
10	5	1	5.00000
11	4	1	5.00000
12	39	3	6.00000
13	37	3	6.00000
14	17	1	6.00000
15	26	2	7.00000
16	10	1	7.00000
17	7	2	7.00000
18	31	2	8.00000
19	27	1	8.00000
20	18	2	8.00000
21	12	2	8.00000
22	35	2	9.00000
23	22	2	9.00000
24	13	2	9.00000
25	6	1	9.00000
26	32	2	10.00000
27	23	2	10.00000
28	9	2	10.00000
29	28	2	11.00000
30	19	2	11.00000
31	14	2	11.00000
32	33	3	13.00000
33	20	3	13.00000
34	8	1	13.00000
35	29	3	14.00000
36	24	3	14.00000
37	25	3	15.00000
38	15	3	15.00000
39	36	3	16.00000

TABLE 22. TRADEOFF FOR FEATURE NUMBER 1

DATE	610	ID	TC1AM1	FEATURE	NUMBER	1	CL	SCORE	0	1	2	3
TRADEOFF	F0R											
	N0	ID	TY									
1	1	1	4	1	0	0	1	3.0000		1		
2	6	6	4	6	0	0	1	4.0000		2		
3	5	5	4	5	0	0	1	4.0000		3		
4	8	8	4	8	0	0	1	5.0000		4		
5	7	7	5	7	0	0	2	5.0000			1	
6	4	4	4	4	0	0	1	5.0000		5		
7	3	3	4	3	0	0	1	5.0000		6		
8	2	2	4	2	0	0	1	5.0000		7		
9	9	9	5	9	0	0	2	6.0000			2	
10	10	10	4	10	0	0	1	7.0000		8		
11	15	15	7	15	0	0	3	8.0000				1
12	14	14	5	14	0	0	2	8.0000			3	
13	13	13	5	13	0	0	2	8.0000			4	
14	12	12	5	12	0	0	2	8.0000			5	
15	11	11	4	11	0	0	1	8.0000		9		
16	16	16	4	16	0	0	1	9.0000		10		
17	20	20	7	20	0	0	3	10.0000				2
18	19	19	5	19	0	0	2	10.0000			6	
19	18	18	5	18	0	0	2	10.0000			7	
20	17	17	4	17	0	0	1	10.0000		11		
21	25	25	7	25	0	0	3	11.0000				3
22	24	24	7	24	0	0	3	11.0000				4
23	23	23	5	23	0	0	2	11.0000			8	
24	22	22	5	22	0	0	2	11.0000			9	
25	21	21	6	21	0	0	3	11.0000				5
26	29	29	7	29	0	0	3	12.0000				6
27	28	28	5	28	0	0	2	12.0000			10	
28	27	27	4	27	0	0	1	12.0000		12		
29	26	26	5	26	0	0	2	12.0000			11	
30	33	33	7	33	0	0	3	13.0000				7
31	32	32	5	32	0	0	2	13.0000			12	
32	31	31	5	31	0	0	2	13.0000			13	
33	30	30	6	30	0	0	3	13.0000				8
34	36	36	7	36	0	0	3	14.0000				9
35	35	35	5	35	0	0	2	14.0000			14	
36	34	34	6	34	0	0	3	14.0000				10
37	37	37	6	37	0	0	3	15.0000				11
38	39	39	6	39	0	0	3	16.0000				12
39	38	38	6	38	0	0	3	16.0000				13

TABLE 23. TRADEOFF FOR FEATURE NUMBER 2

DATE	10	ID	TC1	AM1	FEATURE	NUMBER	2	CL	SCORE	0	1	2	3
TRADEOFF	FOR												
	NO	ID	TY										
1	2	2	4	2	0	0	1	3.0000			1		
2	30	30	6	30	0	0	3	4.0000					1
3	16	16	4	16	0	0	1	4.0000			2		
4	3	3	4	3	0	0	1	4.0000			3		
5	1	1	4	1	0	0	1	4.0000			4		
6	38	38	6	38	0	0	3	5.0000					2
7	34	34	6	34	0	0	3	5.0000					3
8	21	21	6	21	0	0	3	5.0000					4
9	11	11	4	11	0	0	1	5.0000			5		
10	5	5	4	5	0	0	1	5.0000			6		
11	4	4	4	4	0	0	1	5.0000			7		
12	39	39	6	39	0	0	3	6.0000					5
13	37	37	6	37	0	0	3	6.0000					6
14	17	17	4	17	0	0	1	6.0000			8		
15	26	26	5	26	0	0	2	7.0000				1	
16	10	10	4	10	0	0	1	7.0000			9		
17	7	7	5	7	0	0	2	7.0000				2	
18	31	31	5	31	0	0	2	8.0000				3	
19	27	27	4	27	0	0	1	8.0000			10		
20	18	18	5	18	0	0	2	8.0000				4	
21	12	12	5	12	0	0	2	8.0000				5	
22	35	35	5	35	0	0	2	9.0000				6	
23	22	22	5	22	0	0	2	9.0000				7	
24	13	13	5	13	0	0	2	9.0000				8	
25	6	6	4	6	0	0	1	9.0000			11		
26	32	32	5	32	0	0	2	10.0000				9	
27	23	23	5	23	0	0	2	10.0000				10	
28	9	9	5	9	0	0	2	10.0000				11	
29	28	28	5	28	0	0	2	11.0000				12	
30	19	19	5	19	0	0	2	11.0000				13	
31	14	14	5	14	0	0	2	11.0000				14	
32	33	33	7	33	0	0	3	13.0000					7
33	20	20	7	20	0	0	3	13.0000					8
34	8	8	4	8	0	0	1	13.0000			12		
35	29	29	7	29	0	0	3	14.0000					9
36	24	24	7	24	0	0	3	14.0000					10
37	25	25	7	25	0	0	3	15.0000					11
38	15	15	7	15	0	0	3	15.0000					12
39	36	36	7	36	0	0	3	16.0000					13

TABLE 24. STATISTICAL ANALYSIS OUTPUT

DATE 610 ID TC1AM1

STAT

MEAN VECTORS

CLASS	1		
	6.417	10.07	12.62
	6.083	9.143	10.08

DISPERSIONS

CLASS	1	
	48.25	39.50
	39.50	44.25
CLASS	2	
	108.4	92.43
	92.43	85.43
CLASS	3	
	164.5	121.0
	121.0	123.0

OVERALL MEAN

	1	2
	9.795	8.513

COVARIANCES

CLASS	1	
	7.076	.4653
	.4653	7.243
CLASS	2	
	6.923	.3469
	.3469	1.837
CLASS	3	
	5.314	-6.124
	-6.124	21.46

CLASS	1
STANDARD DEVIATIONS	2.660

2.691

CORRELATIONS

	1.000	.6499E-01
	.6499E-01	1.000
CLASS	2	
STANDARD DEVIATIONS	2.631	1.355

CORRELATIONS

	1.000	.9729E-01
	.9729E-01	1.000
CLASS	3	
STANDARD DEVIATIONS	2.305	4.632

CORRELATIONS

	1.000	-.5736
	-.5736	1.000

TABLE 24. STATISTICAL ANALYSIS OUTPUT (CONCLUDED)

```

TOTAL
STANDARD DEVIATIONS
      3.553      3.580

CORRELATIONS
      1.000      .1796
      .1796      1.000

WITHIN
STANDARD DEVIATIONS
      2.537      3.169

CORRELATIONS
      1.000      -.2207
      -.2207      1.000

BETWEEN
STANDARD DEVIATIONS
      2.488      1.666

CORRELATIONS
      1.000      .9793
      .9793      1.000

PAIRWISE DISTANCES
CLASS      1      2      3
      17.31      4.766      7.374
      4.766      .4734      2.710
      7.374      2.710      10.40

CENTER TO MEAN
DISTANCE
CLASS      1      2      3
      4.161      .6881      3.225

DIRECTION COSINES
CLASS      1      2      3
      1.000      -.8609      -.9931
      -.8609      1.000      .7956
      -.9931      .7956      1.000

DIRECTION ANGLES
CLASS      1      2      3
      .0000      149.4      173.3
      149.4      .0000      37.29
      173.3      37.29      .0000

METHOD 1

INVERSE
<1 = 1
<2 = 1
DET = 58.555530294

      1  2  .3648      .6199E-01      .8460E-01
INVERSE
<1 = 1
<2 = 1
DET = 282.83129606

      1  3  -.3233      .1810      .7582E-01

INVERSE
<1 = 1
<2 = 1
DET = 84.558857982

      2  3  1.076      .9239E-01      .3776E-01

```



Columns three and four are the input identification number (ID) and object type number (TY). Columns five, six and seven are information used in sorting and code check and are not important for classifier design. Column eight (CL) is the class number for the vector and is derived by applying the TYPE to CLASS conversion defined by the input "four" card to the values in the column headed "TY". The column headed "SCORE" contains the values of feature number 1. These appear in increasing order, since the ordering was done on this feature. The columns headed "O", "1", "2", "3" relate to sample in classes 1, 2 and 3 and sample vectors that were not assigned a class number (class "O"). Since all of the vectors were given class numbers, there are no entries in column "O". The other columns are to be interpreted in the following way: The three smallest values of feature 1 appear in vectors from class 1. This is shown by the values 1, 2, 3 in column 1. Looking at the column headed "NO", we see that these are vectors 1, 6 and 5 and the score column shows the values of feature 1 to be 3, 4 and 4, respectively. The vector from class 2 with the smallest value for feature 1 is vector number 7 and has the value 5. There are also five vectors from class 1 with this same value, as shown. The listing continues in this way to show how the samples from each class are ordered by feature 1. This particular tradeoff listing shows that the three classes are shuffled together pretty well along feature 1 and that this feature may not be of much help in classifying the data. Table 23, the tradeoff on feature 2, shows that there is some potentially useful structure in this dimension. The listing shows that there are eight vectors from class 1 and six vectors from class 3 that have small values of feature 2. Then, in order, all the vectors from class 2 appear (mixed with three vectors from class 1) having intermediate values of feature 2. Finally, the remaining seven samples from class 3 form a cluster with large values for feature 3 and one maveric from class 1 lies with them. The tradeoff listings can be used in this way to obtain the same kinds of information as can be obtained from a histogram.

Tables 20 and 21 are another kind of tradeoff listing where, instead of listing the order of the samples, the entire feature vector is printed. The heading for Table 20 is "TRADEOFF FOR FEATURE NUMBER -1". The value "1" indicates the feature that is used to order the listing and the minus sign indicates that the listing shows the feature values rather than the class ordinal values. The first four columns of this listing are the same as for the previous listings. The columns headed "1", "2", etc. contain the values for features 1, 2, etc. On this listing, all 10 columns are used, regardless of how many features the samples have.

The listing of feature vectors ordered by the magnitude of feature 2 (see Table 23) showed that the type 6 (assigned to class 3) vectors (points) were of two distinct sub-types; that is, of the 13 type-6 vectors, six had a value of feature 2 between 4 and 6 inclusive, and the other seven had a value of feature 2 between 13 and 16 inclusive. As a result of this discovery, a fourth type was defined (type 7) for the vectors of type 6 with feature 2 values between 13 and 16. The cards for these samples were changed by punching a 7 in field 3. The type column in the listings shows this change.

The classifier portion of the program calculated equations for three lines to be used in separating the data into three classes. It may seem redundant to ask the program to separate data classes where each point has already been defined as belonging to a certain class, but the establishment of boundaries allows the classification of new points, of unknown class, by determining where the new points lie with respect to the previously defined boundaries. It may be said that this first step, this establishment of boundaries, is a learning step, or an assessment of known data, to allow future data to be properly classified.

The class separation lines calculated by the classifier are shown plotted along with the data in Figure 13. [The Method 1 classifier, used for this test case, finds pairwise boundaries; that is, it finds a line to separate a pair of classes, then a line for another pair, etc., until all possible pairs have been treated. For pairwise boundaries between N classes, the number of pairs, and the number of lines, is  $\frac{N(N-1)}{2}$ .] It appears from the figure that these lines do very little in separating classes. Note, however, that, if the pair of classes to be separated is projected onto the normal to the program's calculated line for that pair, the displacement of a projected point along that normal is related to its class. The program calls this displacement a "score" and lists the "score" for every feature vector as part of the tradeoff listing. (This effect is most pronounced in Figure 13 for the class pair 1:2; separation of class 3 is more difficult because it has two parts.) Another view is to say that the optimum lines are parallel to, but displaced from, the calculated pairwise boundaries. By using coefficient cards ("two" cards), these lines may be shifted to become the optimal pairwise boundaries. This is equivalent to adding a biasing value to all "scores", and letting the boundary be equal to a score of zero. The way that the scores are related to the boundary and the sample locations is shown in Figure 14 for an arbitrary boundary line.

A second run of the program was made, with changes to treat the two groups within type 6 as separate. The type 7 was defined as previously discussed, and assigned to class 4 using a class assignment control card ("four" card). The feature vector input cards ("one" cards) for the type 7 vectors were changed to reflect the new type. Thus, there were four types (4, 5, 6, 7) assigned to four classes (1, 2, 3, 4). Again, the program calculated statistics, this time for four classes, listed feature vectors ordered by feature 1 and feature 2, and calculated equations for lines to be used in separating the four classes. These lines are shown in Figure 15. Because of the way the program computes these lines, two of them are not within the scope of Figure 15, and have been represented in the figure by their slopes.



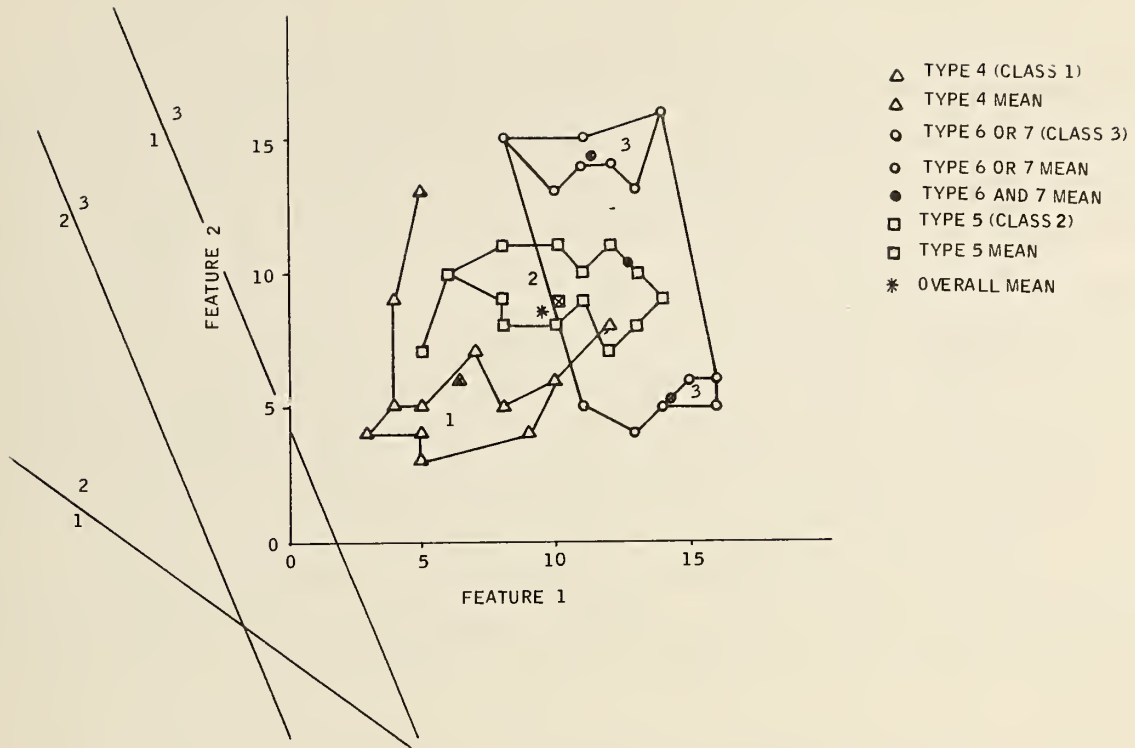


Figure 13. Test Case One Data

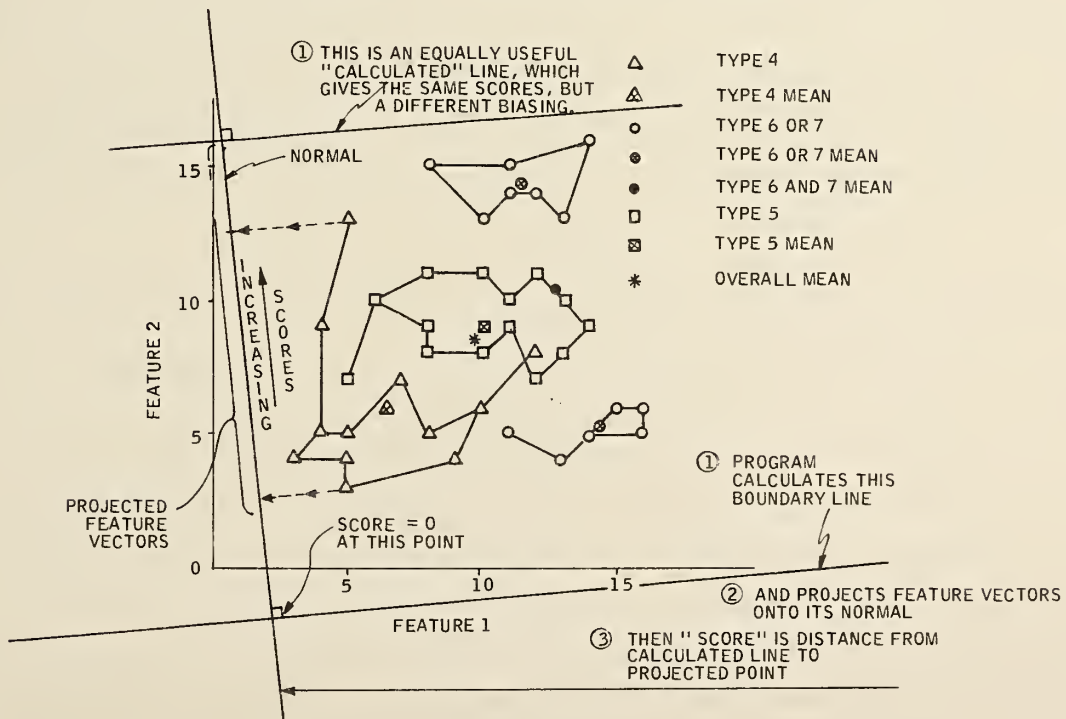


Figure 14. Method of Scoring from Boundary Location

This is adequate to allow finding a normal line to each one, and changes only the relative value of the score for a given feature vector. Determining class membership thus becomes a matter of selecting a threshold, or boundary value of the score with a class above the threshold and another below. It may be seen (Figure 16) that feature 2 itself could be used as a "score", with thresholds at feature 2=6 and feature 2=12. In fact, were it not for the vectors in class 1 (type 4), this classification test case could be solved using only the feature 2 values as scores. The program assigns a larger weighting value to feature 2 (with this data set) because it is a good separator.

It may also be seen that feature 1 could be used to separate classes 1 and 3 if the threshold were chosen at feature 1=10.5; one error would occur in this case (vector 27 would be misplaced in class 3).

The "score" may be thought of as being the dot product of two vectors, namely the feature vector and the vector normal to the program-calculated boundary line. Alternatively, the score is formed by summing a portion of feature 1 plus a portion of feature 2 plus a constant. In either case, by determining a score for a given vector, that vector may be represented by a single value (score) instead of by two values (feature 1 and feature 2).

It is possible to determine several different scores for a given vector by selecting different, non-parallel boundary lines. For example, the boundary between classes 1 and 2 produces a set of scores for the data set, while the boundary between classes 3 and 4 produces another score set. This should not be confused with the phenomenon of many different parallel "boundary" lines between two classes, since in the case of many parallel "boundaries" all the score sets differ only by a constant biasing value (see Figure 14). In other words, any program-calculated boundary defines a score set, by projecting the vectors onto the normal to this program-calculated boundary. In the same manner, any two program-calculated boundaries define two score sets. If the two program-calculated boundaries are parallel, the





score sets are equivalent to each other, but are offset from each other by some bias value. If the two program-calculated boundaries are not parallel, no amount of biasing applied to either score set will make the score sets equivalent.

Thus, a given feature vector may be represented by a single "score" instead of by its component feature values. (This is true for the N-dimensional case as well as for the two-dimensional case under discussion.) Several non-equivalent score sets may initially be required; these can then be treated as "features" and combined to produce a final, single score set which will allow classification of feature vectors by appropriately selecting threshold values (boundary values). In the case under discussion here, the second run produced six pair-wise boundaries, along with a set of 39 scores for each boundary.

The computer runs referred to in the following discussion are the result of processing two additional control card decks, using the data cards that were modified to show type 7 vectors. The discussion of these results is easier when the two runs are merged, thus the complete listings are not given in order, as was done for the run discussed above. The two runs are labelled "DATE 610 ID TC1 BM1" and "DATE 626 ID TC1 CM3", respectively, and this label appears at the top of each page of output. These labels are included in the figures for those who wish to sort them out.

Table 25 is the lead-off listing for the first run. The first few lines of this listing give the control cards that were used, followed by the parameters input from data cards. The LCOUNT and NPCON values are generated by the program and are output for reference at the time they are determined. The next line "EXECUTE" indicates that the "-99" card was encountered. The items listed to the line "FEATURE VECTORS" are values generated by the program in response to input cards. Next follows a listing of the feature vectors as they were read from the input cards and finally a summary of the number of samples assigned to each class.

TABLE 25. PARAMETER AND DATA LISTING

DATE 410 10 TC14M1

+2 01

CROUT = 0

+2 5,6,7

+6 0

+7 =1

+7 =1

+7 1

+7 2

+2 1

PCRT = 1

3 4,12,14,6,7

4 7,0,0,1,1,2,3,4

5 2,1,2

EXECUTE

FLAG(1) = 39

NFV = 39

NC = 4

NT = 7

NF = 2

KMODE = 1

PR98

+3077 +2530 +1528 +1795

TYPE 1 2 3 4 5 6 7

CLAS 0 0 0 1 2 3 4

SELECTED FEATURES

1 2

FEATURE VECTRES

NO 10 TV NF

1 1 4 0 1

2 2 4 0 2

3 3 4 0 3

4 4 4 0 4

5 5 4 0 5

6 6 4 0 6

7 7 4 0 7

8 8 4 0 8

9 9 4 0 9

10 10 4 0 10

11 11 4 0 11

12 12 4 0 12

13 13 4 0 13

14 14 4 0 14

15 15 4 0 15

16 16 4 0 16

17 17 4 0 17

18 18 4 0 18

19 19 4 0 19

20 20 4 0 20

21 21 4 0 21

22 22 4 0 22

23 23 4 0 23

24 24 4 0 24

25 25 4 0 25

26 26 4 0 26

27 27 4 0 27

28 28 4 0 28

29 29 4 0 29

30 30 4 0 30

31 31 4 0 31

32 32 4 0 32

33 33 7 2 33

34 34 6 2 34

35 35 5 2 35

36 36 7 2 36

37 37 6 2 37

38 38 6 2 38

39 39 6 2 39

CL

1

2

3

4

5

6

7

8

9

10

PREP

SELECTED SAMPLES

CLASS 0 1 2 3 4 5 6 7 8 9 10

NUMBER 0 12 14 6 7

NVS = 39

Table 26 is similar to Table 25 with the exception that the pair wise boundaries determined by the previous run have been input as parameter cards (the "2" cards) as shown. These parameters are used to transform the input features and the results of this are listed in the second feature listing.

Near the top of Table 27, below the label "COEFF," the coefficients for the 1, 2 -boundary are listed. The three values given are the additive constant, the feature 1 multiplier and the feature 2 multiplier. The scores are obtained by applying these coefficients to the feature values.

The tradeoff listings of scores for the class pair discriminants show that class membership may be determined by selecting suitable thresholds. This is the first step in using the logical mode classifier. Table 27 shows the scores produced by the data set when projected onto the normal to the discriminant between class 1 and class 2. Note that all class 4 points have scores of 2.0843 or greater, and that any feature vector (point) with a score less than 2.0843 (for this discriminant) is not in class 4. A logical variable may be defined as the score values for this discriminant minus 2.08, in which case all class 4 vectors will show a positive score, and all non-class 4 vectors will show a negative score. This logical variable is shown as a tradeoff listing in Table 28. Score values in Table 28 are different from those in Table 27 because of the way the new variable was defined (must be an integer value) but the phenomenon is clear. A new feature has been defined as: the score from Table 27 minus 2.0617, that difference then multiplied by 1000 to create an integer value with sufficient precision to discriminate. The new integer feature is what is shown as a score in Table 26. In the logical mode, features greater than zero (positive) are called true, while features equal to or less than zero (negative) are called false. Thus, the feature of Table 28 is true only for vectors (points) of class 4. Other logical features may be similarly defined; for example the scores for the discriminant between classes 2 and 3 is shown in Table 29. A new logical variable was defined by subtracting 2.6603 from each score



TABLE 26. LISTING INCLUDING THE TRANSFORMED FEATURES

DATE 626 10 701043  
LC9UNT \* 3 -2 3,3

-3 5,6,7  
-6 0  
-7 1  
-7 2  
-7 3  
5 2,1,2  
2 1,1,0, = 1,636,-7, = 498, -1,5459  
2 2,1,0, = 3,-343, (457, = 1,4755  
2 3,1,0, = 2, 1245, = 1,344, = 3 567  
3 4,2,1,4,6,7  
4 7,1,0, = 1,2,3,4

```
EXECUTE
FLAG(1) • 39
NFV • 39
NC • 4
NT • 7
NF • 2
KNPOE • 3
PR98
•3077 •3590 •1538 •1795
TYPE 1 2 3 4 5 6 7
CLAS 0 0 0 1 2 3 4
SELECTED FEATURES
1 2
FEATURE VECTORS
```

EXTENSION	VELOCITY	1	2	3	4	5	6	7	8	9	10
1	1	1	2	3	4	5	6	7	8	9	10
2	1	1	2	3	4	5	6	7	8	9	10
3	1	1	2	3	4	5	6	7	8	9	10
4	1	1	2	3	4	5	6	7	8	9	10
5	1	1	2	3	4	5	6	7	8	9	10
6	1	1	2	3	4	5	6	7	8	9	10
7	1	1	2	3	4	5	6	7	8	9	10
8	1	1	2	3	4	5	6	7	8	9	10
9	1	1	2	3	4	5	6	7	8	9	10
10	1	1	2	3	4	5	6	7	8	9	10
11	1	1	2	3	4	5	6	7	8	9	10
12	1	1	2	3	4	5	6	7	8	9	10
13	1	1	2	3	4	5	6	7	8	9	10
14	1	1	2	3	4	5	6	7	8	9	10
15	1	1	2	3	4	5	6	7	8	9	10
16	1	1	2	3	4	5	6	7	8	9	10
17	1	1	2	3	4	5	6	7	8	9	10
18	1	1	2	3	4	5	6	7	8	9	10
19	1	1	2	3	4	5	6	7	8	9	10
20	1	1	2	3	4	5	6	7	8	9	10
21	1	1	2	3	4	5	6	7	8	9	10
22	1	1	2	3	4	5	6	7	8	9	10
23	1	1	2	3	4	5	6	7	8	9	10
24	1	1	2	3	4	5	6	7	8	9	10
25	1	1	2	3	4	5	6	7	8	9	10
26	1	1	2	3	4	5	6	7	8	9	10
27	1	1	2	3	4	5	6	7	8	9	10
28	1	1	2	3	4	5	6	7	8	9	10
29	1	1	2	3	4	5	6	7	8	9	10
30	1	1	2	3	4	5	6	7	8	9	10
31	1	1	2	3	4	5	6	7	8	9	10
32	1	1	2	3	4	5	6	7	8	9	10



TABLE 26. LISTING INCLUDING THE TRANSFORMED  
FEATURES (CONCLUDED)

33	33	7	2	33	0	0	0	0	0	13	13	0	0	0	0	0	0	0	0
34	34	6	2	34	0	0	0	0	0	14	5	0	0	0	0	0	0	0	0
35	35	5	2	35	0	0	0	0	0	14	9	0	0	0	0	0	0	0	0
36	36	7	2	36	0	0	0	0	0	14	16	0	0	0	0	0	0	0	0
37	37	6	2	37	0	0	0	0	0	15	6	0	0	0	0	0	0	0	0
38	38	6	2	38	0	0	0	0	0	16	5	0	0	0	0	0	0	0	0
39	39	6	2	39	0	0	0	0	0	16	6	0	0	0	0	0	0	0	0

COEFFICIENT CARDS																			
KT	KT	CP	CONST	COEFF.....															
11000	-1.697			.6198E+01															
21000	.3734			.4517E+01															
31000	-2.728			.3961E+02															

FEATURE VECTORS																			
NO	NO	TY	NO	TY	NO	TY	NO	TY	NO	TY	NO	TY	NO	TY	NO	TY	NO	TY	NO
1	1	4	2	1	0	0	0	0	0	-1172	-151	-793	0	0	0	0	0	0	0
2	2	4	2	2	0	0	0	0	0	-1132	86	-1091	0	0	0	0	0	0	0
3	3	4	2	3	0	0	0	0	0	-1048	-60	-785	0	0	0	0	0	0	0
4	4	4	2	4	0	0	0	0	0	-963	-208	-479	0	0	0	0	0	0	0
5	5	4	2	5	0	0	0	0	0	-1025	-253	-483	0	0	0	0	0	0	0
6	6	4	2	6	0	0	0	0	0	-687	-843	738	0	0	0	0	0	0	0
7	7	5	2	7	0	0	0	0	0	-794	-503	131	0	0	0	0	0	0	0
8	8	4	2	8	0	0	0	0	0	-286	-1388	1965	0	0	0	0	0	0	0
9	9	5	2	9	0	0	0	0	0	-478	-901	1052	0	0	0	0	0	0	0
10	10	4	2	10	0	0	0	0	0	-670	-413	139	0	0	0	0	0	0	0
11	11	4	2	11	0	0	0	0	0	-777	-73	-467	0	0	0	0	0	0	0
12	12	5	2	12	0	0	0	0	0	-523	-515	449	0	0	0	0	0	0	0
13	13	5	2	13	0	0	0	0	0	-439	-662	754	0	0	0	0	0	0	0
14	14	5	2	14	0	0	0	0	0	-270	-958	1366	0	0	0	0	0	0	0
15	15	7	2	15	0	0	0	0	0	68	-1548	2588	0	0	0	0	0	0	0
16	16	4	2	16	0	0	0	0	0	-800	119	-769	0	0	0	0	0	0	0
17	17	4	2	17	0	0	0	0	0	-569	-130	-154	0	0	0	0	0	0	0
18	18	5	2	18	0	0	0	0	0	-399	-425	457	0	0	0	0	0	0	0
19	19	5	2	19	0	0	0	0	0	-146	-868	1374	0	0	0	0	0	0	0
20	20	7	2	20	0	0	0	0	0	23	-1163	1935	0	0	0	0	0	0	0
21	21	6	2	21	0	0	0	0	0	-591	61	-456	0	0	0	0	0	0	0
22	22	5	2	22	0	0	0	0	0	-253	-528	766	0	0	0	0	0	0	0
23	23	5	2	23	0	0	0	0	0	-168	-675	1072	0	0	0	0	0	0	0
24	24	7	2	24	0	0	0	0	0	169	-1266	2294	0	0	0	0	0	0	0
25	25	7	2	25	0	0	0	0	0	254	-1413	2400	0	0	0	0	0	0	0
26	26	5	2	26	0	0	0	0	0	-360	-188	159	0	0	0	0	0	0	0
27	27	4	2	27	0	0	0	0	0	-275	-335	464	0	0	0	0	0	0	0
28	28	5	2	28	0	0	0	0	0	-22	-778	1381	0	0	0	0	0	0	0
29	29	7	2	29	0	0	0	0	0	231	-1220	2298	0	0	0	0	0	0	0
30	30	6	2	30	0	0	0	0	0	-552	299	-753	0	0	0	0	0	0	0
31	31	5	2	31	0	0	0	0	0	-213	-290	468	0	0	0	0	0	0	0
32	32	5	2	32	0	0	0	0	0	-44	-585	1080	0	0	0	0	0	0	0
33	33	7	2	33	0	0	0	0	0	209	-1028	1997	0	0	0	0	0	0	0
34	34	6	2	34	0	0	0	0	0	-405	197	-444	0	0	0	0	0	0	0
35	35	5	2	35	0	0	0	0	0	-67	-393	778	0	0	0	0	0	0	0
36	36	7	2	36	0	0	0	0	0	524	-1425	2918	0	0	0	0	0	0	0
37	37	6	2	37	0	0	0	0	0	-259	94	-134	0	0	0	0	0	0	0
38	38	6	2	38	0	0	0	0	0	-281	287	-436	0	0	0	0	0	0	0
39	39	6	2	39	0	0	0	0	0	-197	139	-130	0	0	0	0	0	0	0

PREP

SELECTED SAMPLES

TABLE 27. TRADEOFF FOR PAIR 1,2

DATE 610 ID TC18M1  
 TRADEOFF FOR  
 PAIR 1 2

COEFF		•36483		•06198		•08459		CL	SCORE	0 1 2 3 4				
	NO	ID	TY											
1	1	1	4	1	0	0	1	1	.8891	1				
2	2	2	4	2	0	0	1	1	.9285	2				
3	3	3	4	3	0	0	1	1	1.0131	3				
4	5	5	4	5	0	0	1	1	1.0357	4				
5	4	4	4	4	0	0	1	1	1.0977	5				
6	16	16	4	16	0	0	1	1	1.2610	6				
7	7	7	5	7	0	0	2	2	1.2669		1			
8	11	11	4	11	0	0	1	1	1.2836	7				
9	6	6	4	6	0	0	1	1	1.3741	8				
10	10	10	4	10	0	0	1	1	1.3908	9				
11	21	21	6	21	0	0	3	3	1.4696			1		
12	17	17	4	17	0	0	1	1	1.4922	10				
13	30	30	6	30	0	0	3	3	1.5089			2		
14	12	12	5	12	0	0	2	2	1.5374		2			
15	9	9	5	9	0	0	2	2	1.5826		3			
16	13	13	5	13	0	0	2	2	1.6220		4			
17	34	34	6	34	0	0	3	3	1.6555			3		
18	18	18	5	18	0	0	2	2	1.6614		5			
19	26	26	5	26	0	0	2	2	1.7007		6			
20	8	8	4	8	0	0	1	1	1.7744	11				
21	38	38	6	38	0	0	3	3	1.7795			4		
22	27	27	4	27	0	0	1	1	1.7853	12				
23	14	14	5	14	0	0	2	2	1.7912		7			
24	37	37	6	37	0	0	3	3	1.8021			5		
25	22	22	5	22	0	0	2	2	1.8079		8			
26	31	31	5	31	0	0	2	2	1.8473		9			
27	39	39	6	39	0	0	3	3	1.8641			6		
28	23	23	5	23	0	0	2	2	1.8925		10			
29	19	19	5	19	0	0	2	2	1.9151		11			
30	35	35	5	35	0	0	2	2	1.9939		12			
31	32	32	5	32	0	0	2	2	2.0165		13			
32	28	28	5	28	0	0	2	2	2.0391		14			
33	20	20	7	20	0	0	4	4	2.0843				1	
34	15	15	7	15	0	0	4	4	2.1295				2	
35	24	24	7	24	0	0	4	4	2.2309				3	
36	33	33	7	33	0	0	4	4	2.2703				4	
37	29	29	7	29	0	0	4	4	2.2929				5	
38	25	25	7	25	0	0	4	4	2.3155				6	
39	36	36	7	36	0	0	4	4	2.5260				7	

TABLE 28. TRADEOFF FOR FEATURE NUMBER 1

DATE	626	ID	TC1CM3	NUMBER	1		SCORE	0	1	2	3	4
TRADEOFF	NO	ID	TY			CL						
1	1	1	4	1	0	0	1	-1172.0000	1			
2	2	2	4	2	0	0	1	-1132.0000	2			
3	3	3	4	3	0	0	1	-1048.0000	3			
4	5	5	4	5	0	0	1	-1025.0000	4			
5	4	4	4	4	0	0	1	-963.0000	5			
6	16	16	4	16	0	0	1	-800.0000	6			
7	7	7	5	7	0	0	2	-794.0000		1		
8	11	11	4	11	0	0	1	-777.0000	7			
9	6	6	4	6	0	0	1	-687.0000	8			
10	10	10	4	10	0	0	1	-670.0000	9			
11	21	21	6	21	0	0	3	-591.0000			1	
12	17	17	4	17	0	0	1	-569.0000	10			
13	30	30	6	30	0	0	3	-552.0000			2	
14	12	12	5	12	0	0	2	-523.0000		2		
15	9	9	5	9	0	0	2	-478.0000		3		
16	13	13	5	13	0	0	2	-439.0000		4		
17	34	34	6	34	0	0	3	-405.0000			3	
18	18	18	5	18	0	0	2	-399.0000		5		
19	26	26	5	26	0	0	2	-360.0000		6		
20	8	8	4	8	0	0	1	-286.0000	11			
21	38	38	6	38	0	0	3	-281.0000			4	
22	27	27	4	27	0	0	1	-275.0000	12			
23	14	14	5	14	0	0	2	-270.0000		7		
24	37	37	6	37	0	0	3	-259.0000			5	
25	22	22	5	22	0	0	2	-253.0000		8		
26	31	31	5	31	0	0	2	-213.0000		9		
27	39	39	6	39	0	0	3	-197.0000			6	
28	23	23	5	23	0	0	2	-168.0000		10		
29	19	19	5	19	0	0	2	-146.0000		11		
30	35	35	5	35	0	0	2	-67.0000		12		
31	32	32	5	32	0	0	2	-44.0000		13		
32	28	28	5	28	0	0	2	-22.0000		14		
33	20	20	7	20	0	0	4	23.0000				1
34	15	15	7	15	0	0	4	68.0000				2
35	24	24	7	24	0	0	4	169.0000				3
36	33	33	7	33	0	0	4	209.0000				4
37	29	29	7	29	0	0	4	231.0000				5
38	25	25	7	25	0	0	4	254.0000				6
39	36	36	7	36	0	0	4	524.0000				7

TABLE 29. TRADEOFF FOR PAIR 2,3

DATE 610 ID TC18M1  
 TRADEOFF FOR  
 PAIR 2 3

COEFF

	2.96373		.04507		-.14755		CL	SCORE	0	1	2	3	4
	NB	ID	TY										
1	15	15	7	15	0	0	4	1.1111					1
2	36	36	7	36	0	0	4	1.2339					2
3	25	25	7	25	0	0	4	1.2463					3
4	8	8	4	8	0	0	1	1.2710		1			
5	24	24	7	24	0	0	4	1.3938					4
6	29	29	7	29	0	0	4	1.4389					5
7	20	20	7	20	0	0	4	1.4963					6
8	33	33	7	33	0	0	4	1.6315					7
9	14	14	5	14	0	0	2	1.7013				1	
10	9	9	5	9	0	0	2	1.7587				2	
11	19	19	5	19	0	0	2	1.7914				3	
12	6	6	4	6	0	0	1	1.8161		2			
13	28	28	5	28	0	0	2	1.8815				4	
14	23	23	5	23	0	0	2	1.9840				5	
15	13	13	5	13	0	0	2	1.9964				6	
16	32	32	5	32	0	0	2	2.0741				7	
17	22	22	5	22	0	0	2	2.1316				8	
18	12	12	5	12	0	0	2	2.1439				9	
19	7	7	5	7	0	0	2	2.1562				10	
20	18	18	5	18	0	0	2	2.2340				11	
21	10	10	4	10	0	0	1	2.2464		3			
22	35	35	5	35	0	0	2	2.2668				12	
23	27	27	4	27	0	0	1	2.3242		4			
24	31	31	5	31	0	0	2	2.3692				13	
25	5	5	4	5	0	0	1	2.4063		5			
26	4	4	4	4	0	0	1	2.4513		6			
27	26	26	5	26	0	0	2	2.4717				14	
28	1	1	4	1	0	0	1	2.5087		7			
29	17	17	4	17	0	0	1	2.5291		8			
30	11	11	4	11	0	0	1	2.5865		9			
31	3	3	4	3	0	0	1	2.5989		10			
32	21	21	6	21	0	0	3	2.7217				1	
33	2	2	4	2	0	0	1	2.7464		11			
34	37	37	6	37	0	0	3	2.7545				2	
35	16	16	4	16	0	0	1	2.7792		12			
36	39	39	6	39	0	0	3	2.7995				3	
37	34	34	6	34	0	0	3	2.8569				4	
38	38	38	6	38	0	0	3	2.9471				5	
39	30	30	6	30	0	0	3	2.9594				6	



(and multiplying by 1000 to produce integers); it is shown in Table 30 and is true for feature vectors in class 3 (with two errors; unfortunately this seems the best separation of class 3 that is possible). The third variable to be used for the logical mode classifier was formed from the scores on the discriminant between class 2 and class 4 (see Table 31) by placing a threshold at 1.3349 (see Table 32).

Having defined these three logical variables, which the program will call "true" if they are positive and "false" if not, a table may be constructed showing all possible combinations of these three logical variables and the resulting class. It is shown in Table 33. While all states (combinations) in Table 33 are logically possible, four of the states do not occur for this data set with the thresholds selected as described. For example, no single feature vector will produce a logically true value in all three score listings, because of the data distribution and threshold selections. The other "disallowed" states happen for the same reason: for instance, no feature vector has logical variable 1 as false when logical variables 2 and 3 are true.

The errors in classes 2 and 3 occur because of the imperfections in the discriminants calculated by the method 1 classifier. This is a fault of the data distribution and not of the program itself. This may be partially corrected by defining additional logical variables (up to a total of 10) using the same technique as before; however, each new logical variable doubles the number of logically possible states in Table 33. One of these additional logical variables could be defined by placing a threshold at 1.875 in Table 27. This separates five vectors in class 2 from the rest of the data set, and would produce eight new possible states in Table 33. There would then be class 2 vectors in more than a single state of Table 33; these states then would need to be combined in some manner to give a single "output" for any class 2 vector. The technique of combining logical states is discussed elsewhere, in the section concerning Karnaugh mapping.

The classifier program produces a unique number for each logical state of



TABLE 30. TRADEOFF FOR FEATURE NUMBER 2

DATE	626	ID	TC1CM3	TRADEOFF FOR	FEATURE	NUMBER	2	CL	SCORE	0	1	2	3	4
		N0	ID	TY										
1	15	15	7	15	0	0	4	-1548.0000						1
2	36	36	7	36	0	0	4	-1425.0000						2
3	25	25	7	25	0	0	4	-1413.0000						3
4	8	8	4	8	0	0	1	-1388.0000		1				
5	24	24	7	24	0	0	4	-1266.0000						4
6	29	29	7	29	0	0	4	-1220.0000						5
7	20	20	7	20	0	0	4	-1163.0000						6
8	33	33	7	33	0	0	4	-1028.0000						7
9	14	14	5	14	0	0	2	-958.0000				1		
10	9	9	5	9	0	0	2	-911.0000				2		
11	19	19	5	19	0	0	2	-868.0000				3		
12	6	6	4	6	0	0	1	-843.0000		2				
13	28	28	5	28	0	0	2	-778.0000				4		
14	23	23	5	23	0	0	2	-675.0000				5		
15	13	13	5	13	0	0	2	-663.0000				6		
16	32	32	5	32	0	0	2	-585.0000				7		
17	22	22	5	22	0	0	2	-528.0000				8		
18	12	12	5	12	0	0	2	-515.0000				9		
19	7	7	5	7	0	0	2	-503.0000				10		
20	18	18	5	18	0	0	2	-425.0000				11		
21	10	10	4	10	0	0	1	-413.0000		3				
22	35	35	5	35	0	0	2	-393.0000				12		
23	27	27	4	27	0	0	1	-335.0000		4				
24	31	31	5	31	0	0	2	-290.0000				13		
25	5	5	4	5	0	0	1	-253.0000		5				
26	4	4	4	4	0	0	1	-208.0000		6				
27	26	26	5	26	0	0	2	-188.0000				14		
28	1	1	4	1	0	0	1	-151.0000		7				
29	17	17	4	17	0	0	1	-130.0000		8				
30	11	11	4	11	0	0	1	-73.0000		9				
31	3	3	4	3	0	0	1	-60.0000		10				
32	21	21	6	21	0	0	3	61.0000				1		
33	2	2	4	2	0	0	1	86.0000		11				
34	37	37	6	37	0	0	3	94.0000				2		
35	16	16	4	16	0	0	1	119.0000		12				
36	39	39	6	39	0	0	3	139.0000					3	
37	34	34	6	34	0	0	3	197.0000					4	
38	38	38	6	38	0	0	3	287.0000					5	
39	30	30	6	30	0	0	3	299.0000					6	

TABLE 31. TRADEOFF FOR PAIR 2,4

DATE 610 ID TC1BM1  
 TRADEOFF FOR  
 PAIR 2 4

COEFF

•.69355		•00396		•30567		CL	SCORE	0 1 2 3 4				
N0	ID	TY										
1	2	2	4	2	0	0	1	.2433	1			
2	1	1	4	1	0	0	1	.5410	2			
3	3	3	4	3	0	0	1	.5489	3			
4	16	16	4	16	0	0	1	.5648	4			
5	30	30	6	30	0	0	3	.5806			1	
6	5	5	4	5	0	0	1	.8507	5			
7	4	4	4	4	0	0	1	.8546	6			
8	11	11	4	11	0	0	1	.8665	7			
9	21	21	6	21	0	0	3	.8784			2	
10	34	34	6	34	0	0	3	.8902			3	
11	38	38	6	38	0	0	3	.8982			4	
12	17	17	4	17	0	0	1	1.1801	8			
13	37	37	6	37	0	0	3	1.1999			5	
14	39	39	6	39	0	0	3	1.2038			6	
15	7	7	5	7	0	0	2	1.4660		1		
16	10	10	4	10	0	0	1	1.4739	9			
17	26	26	5	26	0	0	2	1.4937		2		
18	12	12	5	12	0	0	2	1.7835		3		
19	18	18	5	18	0	0	2	1.7914		4		
20	27	27	4	27	0	0	1	1.7993	10			
21	31	31	5	31	0	0	2	1.8033		5		
22	6	6	4	6	0	0	1	2.0733	11			
23	13	13	5	13	0	0	2	2.0892		6		
24	22	22	5	22	0	0	2	2.1011		7		
25	35	35	5	35	0	0	2	2.1129		8		
26	9	9	5	9	0	0	2	2.3869		9		
27	23	23	5	23	0	0	2	2.4067		10		
28	32	32	5	32	0	0	2	2.4147		11		
29	14	14	5	14	0	0	2	2.7005		12		
30	19	19	5	19	0	0	2	2.7084		13		
31	28	28	5	28	0	0	2	2.7164		14		
32	8	8	4	8	0	0	1	3.3000	12			
33	20	20	7	20	0	0	4	3.3198			1	
34	33	33	7	33	0	0	4	3.3317			2	
35	24	24	7	24	0	0	4	3.6294			3	
36	29	29	7	29	0	0	4	3.6334			4	
37	15	15	7	15	0	0	4	3.9232			5	
38	25	25	7	25	0	0	4	3.9351			6	
39	36	36	7	36	0	0	4	4.2527			7	

TABLE 32. TRADEOFF FOR FEATURE NUMBER 3

DATE	626	ID	TC1CM3	TRADEOFF	FOR	FEATURE	NUMBER	3	CL	SCORE	0	1	2	3	4
-	N0	ID	TY												
1	2	2	4	2	0	0	1	0	1	-1091.0000		1			
2	1	1	4	1	0	0	1	0	1	-793.0000		2			
3	3	3	4	3	0	0	1	0	1	-785.0000		3			
4	16	16	4	16	0	0	1	0	1	-769.0000		4			
5	30	30	6	30	0	0	3	0	3	-753.0000				1	
6	5	5	4	5	0	0	1	0	1	-483.0000		5			
7	4	4	4	4	0	0	1	0	1	-479.0000		6			
8	11	11	4	11	0	0	1	0	1	-467.0000		7			
9	21	21	6	21	0	0	3	0	3	-456.0000				2	
10	34	34	6	34	0	0	3	0	3	-444.0000				3	
11	38	38	6	38	0	0	3	0	3	-436.0000				4	
12	17	17	4	17	0	0	1	0	1	-154.0000		8			
13	37	37	6	37	0	0	3	0	3	-134.0000				5	
14	39	39	6	39	0	0	3	0	3	-130.0000				6	
15	7	7	5	7	0	0	2	0	2	131.0000			1		
16	10	10	4	10	0	0	1	0	1	139.0000		9			
17	26	26	5	26	0	0	2	0	2	159.0000			2		
18	12	12	5	12	0	0	2	0	2	449.0000			3		
19	18	18	5	18	0	0	2	0	2	457.0000			4		
20	27	27	4	27	0	0	1	0	1	464.0000		10			
21	31	31	5	31	0	0	2	0	2	468.0000			5		
22	6	6	4	6	0	0	1	0	1	738.0000		11			
23	13	13	5	13	0	0	2	0	2	754.0000			6		
24	22	22	5	22	0	0	2	0	2	766.0000			7		
25	35	35	5	35	0	0	2	0	2	778.0000			8		
26	9	9	5	9	0	0	2	0	2	1052.0000			9		
27	23	23	5	23	0	0	2	0	2	1072.0000			10		
28	32	32	5	32	0	0	2	0	2	1080.0000			11		
29	14	14	5	14	0	0	2	0	2	1366.0000			12		
30	19	19	5	19	0	0	2	0	2	1374.0000			13		
31	28	28	5	28	0	0	2	0	2	1381.0000			14		
32	8	8	4	8	0	0	1	0	1	1965.0000		12			
33	20	20	7	20	0	0	4	0	4	1985.0000					1
34	33	33	7	33	0	0	4	0	4	1997.0000					2
35	24	24	7	24	0	0	4	0	4	2294.0000					3
36	29	29	7	29	0	0	4	0	4	2298.0000					4
37	15	15	7	15	0	0	4	0	4	2588.0000					5
38	25	25	7	25	0	0	4	0	4	2600.0000					6
39	36	36	7	36	0	0	4	0	4	2918.0000					7

TABLE 33. POSSIBLE COMBINATIONS OF THREE LOGICAL VARIABLES

<u>Logical Variable 1</u>	<u>Logical Variable 2</u>	<u>Logical Variable 3</u>	<u>Class</u>
True	True	True	State does not occur
True	True	False	State does not occur
True	False	True	4
True	False	False	State does not occur
False	True	True	State does not occur
False	True	False	3 (with 2 errors)
False	False	True	2 (with 4 errors)
False	False	False	1

Table 33 by substituting a digit "1" for the value "true," and digit "0" for the value "false." The table then appears as shown below.

<u>Logical Variable 1</u>	<u>Logical Variable 2</u>	<u>Logical Variable 3</u>	<u>Octal Number</u>	<u>Class</u>
1	1	1	7	D. N. O.
1	1	0	6	D. N. O.
1	0	1	5	4
1	0	0	4	D. N. O.
0	1	1	3	D. N. O.
0	1	0	2	3 (with errors)
0	0	1	1	2 (with errors)
0	0	0	0	1

For the listing shown in Table 34, the logical variables are combined into an octal number, as shown in the table above.

TABLE 34. LOGICAL TRADEOFF FOR TEST CASE 1

DATE 626 ID TC1CM3  
LCBUNT = 3

	NO	ID	TY			CL	SCORE	0	1	2	3	4
1	17	17	4	17	0	0	00000000		1			
2	11	11	4	11	0	0	00000000		2			
3	5	5	4	5	0	0	00000000		3			
4	4	4	4	4	0	0	00000000		4			
5	3	3	4	3	0	0	00000000		5			
6	1	1	4	1	0	0	00000000		6			
7	35	35	5	35	0	0	00000001			1		
8	32	32	5	32	0	0	00000001			2		
9	31	31	5	31	0	0	00000001			3		
10	28	28	5	28	0	0	00000001			4		
11	27	27	4	27	0	0	00000001	7				
12	26	26	5	26	0	0	00000001			5		
13	23	23	5	23	0	0	00000001			6		
14	22	22	5	22	0	0	00000001			7		
15	19	19	5	19	0	0	00000001			8		
16	18	18	5	18	0	0	00000001			9		
17	14	14	5	14	0	0	00000001			10		
18	13	13	5	13	0	0	00000001			11		
19	12	12	5	12	0	0	00000001			12		
20	10	10	4	10	0	0	00000001	8				
21	9	9	5	9	0	0	00000001			13		
22	8	8	4	8	0	0	00000001	9				
23	7	7	5	7	0	0	00000001			14		
24	6	6	4	6	0	0	00000001	10				
25	39	39	6	39	0	0	00000002				1	
26	38	38	6	38	0	0	00000002				2	
27	37	37	6	37	0	0	00000002				3	
28	34	34	6	34	0	0	00000002				4	
29	30	30	6	30	0	0	00000002				5	
30	21	21	6	21	0	0	00000002				6	
31	16	16	4	16	0	0	00000002	11				
32	2	2	4	2	0	0	00000002	12				
33	36	36	7	36	0	0	00000005					1
34	33	33	7	33	0	0	00000005					2
35	29	29	7	29	0	0	00000005					3
36	25	25	7	25	0	0	00000005					4
37	24	24	7	24	0	0	00000005					5
38	20	20	7	20	0	0	00000005					6
39	15	15	7	15	0	0	00000005					7



## TEST CASE 2

A second test case was developed; the data from test case one was modified by including a third feature for each feature vector (data point), thus providing a three-dimensional data set for classification.

Two program runs were made initially, one using the method 1 classifier (Table 35) and another using method 2 (Table 36). The pairwise class boundaries were not the same for the two methods, but the errors were approximately equal. One class was not separated into two or more subclasses as had been done in the two-dimensional test case, since the additional feature in the three-dimensional case tended to unify the class which had appeared split in two dimensions, as shown by the scores determined for each pairwise class boundary.

Using the pairwise class boundary scores from the method 1 and method 2 runs, with suitable threshold values, the method 3 (logical mode) classifier was executed with six logical variables. The result was ten distinct groups of features (out of  $2^6 = 64$  possible groups) with a unique octal score for each group. (Octal is a number system with eight as its base, instead of a base of 10, as in the decimal system.) The controls and results are shown in Table 37.

The three classes were divided among these ten groups in such a way that seven of the groups contained vectors of a single class, while in three of the groups, more than one class was represented. By combining these groups, the three original classes may be recovered; any subsequent vector is then assigned an octal score which determines which group it is in, thus also assigning the vector to its proper class. The task of combining these 10 groups logically into the original classes is facilitated by a technique called Karnaugh mapping, which will be discussed briefly.

TABLE 35. TEST CASE 2 PROGRAM OUTPUTS, MODE 1

```

DATE 511 10 TESCA2A1
LCQUANT = 0
PCOB = 0
EXECUTE
FLAG(1) = 39
NFV = 39
NC = 2
NT = 6
NF = 3
KMSOE = 1
PRRB
+3077 +3590 +3333
TYPE 1 2 3 4 5 6
CLAS 0 0 0 1 2 3
SELECTED FEATURES
1 2 3
FEATURE VECTORS
NR 10 TV NF
1 1 4 2 1
2 2 4 3 2
3 3 4 3 3
4 4 4 3 4
5 5 4 3 5
6 6 4 3 6
7 7 4 3 7
8 8 4 3 8
9 9 4 3 9
10 10 4 3 10
11 11 4 3 11
12 12 4 3 12
13 13 4 3 13
14 14 4 3 14
15 15 4 3 15
16 16 4 3 16
17 17 4 3 17
18 18 4 3 18
19 19 4 3 19
20 20 4 3 20
21 21 4 3 21
22 22 4 3 22
23 23 4 3 23
24 24 4 3 24
25 25 4 3 25
26 26 4 3 26
27 27 4 3 27
28 28 4 3 28
29 29 4 3 29
30 30 4 3 30
31 31 4 3 31
32 32 4 3 32
33 33 4 3 33
34 34 4 3 34
35 35 4 3 35
36 36 4 3 36
37 37 4 3 37
38 38 4 3 38
39 39 4 3 39
PRR
SELECTED SAMPLES
CLASS 1 2 3 4 5 6 7 8 9 10
NUMBER 12 14 15
VSS = 39

```

TABLE 35. TEST CASE 2 PROGRAM OUTPUTS, MODE 1 (CONTINUED)

DATE 511 ID TESCA2A1  
 TRADEOFF FOR  
 PAIR 1 2

COEFF		.09475		.06771		.09176		.02103		SCORE				
	NO	ID	TY					CL			0	1	2	3
1	1	1	4	1	0	0	1	1	.9173		1			
2	2	2	4	2	0	0	1	1	1.0030		2			
3	3	3	4	3	0	0	1	1	1.0317		3			
4	5	5	4	5	0	0	1	1	1.0347		4			
5	4	4	4	4	0	0	1	1	1.0814		5			
6	16	16	4	16	0	0	1	1	1.1763		6			
7	7	7	5	7	0	0	2	2	1.2859			1		
8	10	10	4	10	0	0	1	1	1.3372		7			
9	11	11	4	11	0	0	1	1	1.3476		8			
10	6	6	4	6	0	0	1	1	1.3597		9			
11	12	12	5	12	0	0	2	2	1.4336			2		
12	17	17	4	17	0	0	1	1	1.4906	10				
13	13	13	5	13	0	0	2	2	1.5674			3		
14	18	18	5	18	0	0	2	2	1.5900			4		
15	9	9	5	9	0	0	2	2	1.6079			5		
16	21	21	6	21	0	0	3	3	1.6348				1	
17	30	30	6	30	0	0	3	3	1.6364				2	
18	8	8	4	8	0	0	1	1	1.7103	11				
19	26	26	5	26	0	0	2	2	1.7178		6			
20	27	27	4	27	0	0	1	1	1.7675	12				
21	22	22	5	22	0	0	2	2	1.7916		7			
22	38	38	6	38	0	0	3	3	1.8051				3	
23	34	34	6	34	0	0	3	3	1.8169				4	
24	14	14	5	14	0	0	2	2	1.8561		8			
25	31	31	5	31	0	0	2	2	1.8563		9			
26	37	37	6	37	0	0	3	3	1.8712				5	
27	39	39	6	39	0	0	3	3	1.8759				6	
28	23	23	5	23	0	0	2	2	1.9464		10			
29	35	35	5	35	0	0	2	2	1.9947		11			
30	32	32	5	32	0	0	2	2	1.9977		12			
31	20	20	6	20	0	0	3	3	2.0278				7	
32	19	19	5	19	0	0	2	2	2.0336		13			
33	28	28	5	28	0	0	2	2	2.0638		14			
34	15	15	6	15	0	0	3	3	2.1180				8	
35	24	24	6	24	0	0	3	3	2.2714				9	
36	29	29	6	29	0	0	3	3	2.2970				10	
37	25	25	6	25	0	0	3	3	2.4683				11	
38	33	33	6	33	0	0	3	3	2.4833				12	
39	36	36	6	36	0	0	3	3	2.7211				13	

TABLE 35. TEST CASE 2 PROGRAM OUTPUTS. MODE 1 (CONTINUED)

DATE 511 ID TESCA2A1  
 TRADEOFF FOR  
 PAIR 1 3

COEFF		.17126		.09952		.06737		SCORE	C	1	2	3
-1.04813	NB	ID	TY			CL						
1	1	1	4	1	0	0	1	.6721		1		
2	5	5	4	5	0	0	1	.8781		2		
3	4	4	4	4	0	0	1	.9120		3		
4	3	3	4	3	0	0	1	.9472		4		
5	2	2	4	2	0	0	1	1.0498		5		
6	6	6	4	6	0	0	1	1.0715		6		
7	7	7	5	7	0	0	2	1.1784			1	
8	16	16	4	16	0	0	1	1.2281		7		
9	10	10	4	10	0	0	1	1.2515		8		
10	12	12	5	12	0	0	2	1.3202			2	
11	8	8	4	8	0	0	1	1.3714		9		
12	13	13	5	13	0	0	2	1.5544			3	
13	9	9	5	9	0	0	2	1.5809			4	
14	11	11	4	11	0	0	1	1.6279	10			
15	18	18	5	18	0	0	2	1.7300			5	
16	17	17	4	17	0	0	1	1.8005	11			
17	14	14	5	14	0	0	2	2.0903			6	
18	22	22	5	22	0	0	2	2.1355			7	
19	15	15	6	15	0	0	3	2.1515				1
20	20	20	6	20	0	0	3	2.1603				2
21	27	27	4	27	0	0	1	2.2073	12			
22	26	26	5	26	0	0	2	2.2425		8		
23	21	21	6	21	0	0	3	2.4111				3
24	23	23	5	23	0	0	2	2.4372		9		
25	31	31	5	31	0	0	2	2.4459		10		
26	32	32	5	32	0	0	2	2.5102		11		
27	30	30	6	30	0	0	3	2.5194				4
28	19	19	5	19	0	0	2	2.5675		12		
29	28	28	5	28	0	0	2	2.5732		13		
30	35	35	5	35	0	0	2	2.6493		14		
31	24	24	6	24	0	0	3	2.7005				5
32	38	38	6	38	0	0	3	2.7285				6
33	29	29	6	29	0	0	3	2.7370				7
34	39	39	6	39	0	0	3	2.7606				8
35	37	37	6	37	0	0	3	2.7915				9
36	34	34	6	34	0	0	3	2.8575				10
37	25	25	6	25	0	0	3	3.1369				11
38	33	33	6	33	0	0	3	3.4824				12
39	36	36	6	36	0	0	3	3.6154				13

TABLE 35. TEST CASE 2 PROGRAM OUTPUTS, MODE 1 (CONCLUDED)

DATE 511 ID TESCA2A1  
 TRADEOFF FOR  
 PAIR 2 3

COEFF							SCORE				
.67539		.08413		.04786		.04697		0	1	2	3
NO		ID	TY			CL					
1	1	1	4	1	0	0	1	1.6828	1		
2	5	5	4	5	0	0	1	1.7209	2		
3	4	4	4	4	0	0	1	1.7581	3		
4	3	3	4	3	0	0	1	1.8041	4		
5	6	6	4	6	0	0	1	1.8184	5		
6	16	16	4	16	0	0	1	1.8589	6		
7	12	12	5	12	0	0	2	1.8722		1	
8	10	10	4	10	0	0	1	1.8811	7		
9	2	2	4	2	0	0	1	1.8972	8		
10	7	7	5	7	0	0	2	1.9007		2	
11	8	8	4	8	0	0	1	1.9061	9		
12	13	13	5	13	0	0	2	2.0140		3	
13	9	9	5	9	0	0	2	2.0815		4	
14	18	18	5	18	0	0	2	2.0874		5	
15	11	11	4	11	0	0	1	2.1514	10		
16	17	17	4	17	0	0	1	2.1796	11		
17	20	20	6	20	0	0	3	2.2798			1
18	15	15	6	15	0	0	3	2.3011			2
19	22	22	5	22	0	0	2	2.3134		6	
20	14	14	5	14	0	0	2	2.3446		7	
21	27	27	4	27	0	0	1	2.3497	12		
22	26	26	5	26	0	0	2	2.3957		8	
23	31	31	5	31	0	0	2	2.4808		9	
24	32	32	5	32	0	0	2	2.4825	10		
25	23	23	5	23	0	0	2	2.5021	11		
26	28	28	5	28	0	0	2	2.5402	12		
27	35	35	5	35	0	0	2	2.5658	13		
28	29	29	6	29	0	0	3	2.5898			3
29	21	21	6	21	0	0	3	2.5917			4
30	24	24	6	24	0	0	3	2.5996			5
31	19	19	5	19	0	0	2	2.6068	14		
32	30	30	6	30	0	0	3	2.6181			6
33	38	38	6	38	0	0	3	2.6366			7
34	39	39	6	39	0	0	3	2.6374			8
35	37	37	6	37	0	0	3	2.6942			9
36	34	34	6	34	0	0	3	2.7971			10
37	25	25	6	25	0	0	3	2.8823			11
38	36	36	6	36	0	0	3	3.0887			12
39	33	33	6	33	0	0	3	3.0958			13



TABLE 36. TEST CASE 2 PROGRAM OUTPUTS, MODE 2

DATE 511 ID TESCAS2A

LCOUNT • 0

EA  
-2 2  
-3 5, 6, 7  
-6 0  
-7 1  
-7 2  
-7 3  
-8 0

NPCBA • 3

3 3,12,14,13  
4 6,0,0,0,1,2,3  
5 3,1,2,3

EXECUTE

EXECUTE  
FLAG(1) • 39

✓FV \* 39

VC ■ 3

VT • 6  
15 • 8

✓F ■ 3  
KMADE ■ 2

KMBDE  
PRBB

•3077 •3590 •3333

TYPE	1	2	3	4	5	6
------	---	---	---	---	---	---

CLAS 0 0 0 1 2 3

### SELECTED FEATURES

1	2	3
FEATURE		

FEATURE	VECTORS	10	TY	NP	CL	1	2	3	4	5	6	7	8	9	10
1	1	4	3	1	0	0	0	0	0	0	0	0	0	0	0
2	2	4	3	2	0	0	0	0	0	0	0	0	0	0	0
3	3	4	3	3	0	0	0	0	0	0	0	0	0	0	0
4	4	4	3	4	0	0	0	0	0	0	0	0	0	0	0
5	5	4	3	5	0	0	0	0	0	0	0	0	0	0	0
6	6	4	3	6	0	0	0	0	0	0	0	0	0	0	0
7	7	4	3	7	0	0	0	0	0	0	0	0	0	0	0
8	8	4	3	8	0	0	0	0	0	0	0	0	0	0	0
9	9	4	3	9	0	0	0	0	0	0	0	0	0	0	0
10	10	4	3	10	0	0	0	0	0	0	0	0	0	0	0
11	11	4	3	11	0	0	0	0	0	0	0	0	0	0	0
12	12	4	3	12	0	0	0	0	0	0	0	0	0	0	0
13	13	4	3	13	0	0	0	0	0	0	0	0	0	0	0
14	14	4	3	14	0	0	0	0	0	0	0	0	0	0	0
15	15	4	3	15	0	0	0	0	0	0	0	0	0	0	0
16	16	4	3	16	0	0	0	0	0	0	0	0	0	0	0
17	17	4	3	17	0	0	0	0	0	0	0	0	0	0	0
18	18	4	3	18	0	0	0	0	0	0	0	0	0	0	0
19	19	4	3	19	0	0	0	0	0	0	0	0	0	0	0
20	20	4	3	20	0	0	0	0	0	0	0	0	0	0	0
21	21	4	3	21	0	0	0	0	0	0	0	0	0	0	0
22	22	4	3	22	0	0	0	0	0	0	0	0	0	0	0
23	23	4	3	23	0	0	0	0	0	0	0	0	0	0	0
24	24	4	3	24	0	0	0	0	0	0	0	0	0	0	0
25	25	4	3	25	0	0	0	0	0	0	0	0	0	0	0
26	26	4	3	26	0	0	0	0	0	0	0	0	0	0	0
27	27	4	3	27	0	0	0	0	0	0	0	0	0	0	0
28	28	4	3	28	0	0	0	0	0	0	0	0	0	0	0
29	29	4	3	29	0	0	0	0	0	0	0	0	0	0	0
30	30	4	3	30	0	0	0	0	0	0	0	0	0	0	0
31	31	4	3	31	0	0	0	0	0	0	0	0	0	0	0
32	32	4	3	32	0	0	0	0	0	0	0	0	0	0	0
33	33	4	3	33	0	0	0	0	0	0	0	0	0	0	0
34	34	4	3	34	0	0	0	0	0	0	0	0	0	0	0
35	35	4	3	35	0	0	0	0	0	0	0	0	0	0	0
36	36	4	3	36	0	0	0	0	0	0	0	0	0	0	0
37	37	4	3	37	0	0	0	0	0	0	0	0	0	0	0
38	38	4	3	38	0	0	0	0	0	0	0	0	0	0	0
39	39	4	3	39	0	0									

PREP

SELECTED SAMPLES

CLASS	1	2	3	4	5	6	7	8	9	10
NUMBER	12	14	13							

SS • 39

TABLE 36. TEST CASE 2 PROGRAM OUTPUTS, MODE 2 (CONTINUED)

DATE 511 ID TESCAS2A  
 TRADEOFF FOR  
 PAIR 1 2

COEFF		.09819		.30950		.19911		SCORE				
-3.25505	NO	ID	TY			CL			0	1	2	3
1	16	16	4	16	0	0	1	-.1378		1		
2	4	4	4	4	0	0	1	.5754		2		
3	12	12	5	12	0	0	2	.6038			1	
4	3	3	4	3	0	0	1	.6641		3		
5	1	1	4	1	0	0	1	.6668		4		
6	5	5	4	5	0	0	1	.6763		5		
7	10	10	4	10	0	0	1	.7934		6		
8	2	2	4	2	0	0	1	.9519		7		
9	18	18	5	18	0	0	2	.9993			2	
10	17	17	4	17	0	0	1	1.1767		8		
11	13	13	5	13	0	0	2	1.3115			3	
12	7	7	5	7	0	0	2	1.3935			4	
13	38	38	6	38	0	0	3	1.4564				1
14	11	11	4	11	0	0	1	1.4673		9		
15	6	6	4	6	0	0	1	1.5161		10		
16	39	39	6	39	0	0	3	1.5667				2
17	27	27	4	27	0	0	1	1.5939		11		
18	26	26	5	26	0	0	2	1.6826			5	
19	22	22	5	22	0	0	2	1.8052			6	
20	31	31	5	31	0	0	2	1.8912			7	
21	30	30	6	30	0	0	3	2.0470				3
22	8	8	4	8	0	0	1	2.0558		12		
23	37	37	6	37	0	0	3	2.0659				4
24	35	35	5	35	0	0	2	2.0998			8	
25	32	32	5	32	0	0	2	2.1120			9	
26	9	9	5	9	0	0	2	2.2211		10		
27	20	20	6	20	0	0	3	2.3477				5
28	21	21	6	21	0	0	3	2.5583				6
29	34	34	6	34	0	0	3	2.6538				7
30	23	23	5	23	0	0	2	2.7120			11	
31	28	28	5	28	0	0	2	2.7215			12	
32	14	14	5	14	0	0	2	2.9261			13	
33	15	15	6	15	0	0	3	3.1685				8
34	29	29	6	29	0	0	3	3.2518				9
35	19	19	5	19	0	0	2	3.5207		14		
36	24	24	6	24	0	0	3	3.5518				10
37	25	25	6	25	0	0	3	4.8568				11
38	33	33	6	33	0	0	3	5.0316				12
39	36	36	6	36	0	0	3	5.0627				13

TABLE 36. TEST CASE 2 PROGRAM OUTPUTS, MODE 2 (CONTINUED)

DATE 511 ID TESCAS2A  
 TRADEOFF FOR  
 PAIR 1 3

COEFF

.97587		.05391		.03424		.02856		SCORE	0	1	2	3
NO	ID	TY				CL						
1	1	1	4	1	0	0	1	1.6173		1		
2	5	5	4	5	0	0	1	1.6483		2		
3	4	4	4	4	0	0	1	1.6737		3		
4	3	3	4	3	0	0	1	1.6966		4		
5	6	6	4	6	0	0	1	1.7282		5		
6	16	16	4	16	0	0	1	1.7408		6		
7	2	2	4	2	0	0	1	1.7480		7		
8	10	10	4	10	0	0	1	1.7643		8		
9	12	12	5	12	0	0	2	1.7667			1	
10	7	7	5	7	0	0	2	1.7707			2	
11	8	8	4	8	0	0	1	1.8048		9		
12	13	13	5	13	0	0	2	1.8581			3	
13	9	9	5	9	0	0	2	1.8988			4	
14	18	18	5	18	0	0	2	1.9031			5	
15	11	11	4	11	0	0	1	1.9211	10			
16	17	17	4	17	0	0	1	1.9489	11			
17	20	20	6	20	0	0	3	2.0457				1
18	22	22	5	22	0	0	2	2.0484			6	
19	15	15	6	15	0	0	3	2.0635				2
20	27	27	4	27	0	0	1	2.0680	12			
21	14	14	5	14	0	0	2	2.0694			7	
22	26	26	5	26	0	0	2	2.0909			8	
23	31	31	5	31	0	0	2	2.1505			9	
24	32	32	5	32	0	0	2	2.1619			10	
25	23	23	5	23	0	0	2	2.1683			11	
26	21	21	6	21	0	0	3	2.1971				3
27	28	28	5	28	0	0	2	2.1993			12	
28	35	35	5	35	0	0	2	2.2101			13	
29	30	30	6	30	0	0	3	2.2135				4
30	19	19	5	19	0	0	2	2.2343		14		
31	38	38	6	38	0	0	3	2.2381				5
32	39	39	6	39	0	0	3	2.2438				6
33	29	29	6	29	0	0	3	2.2449				7
34	24	24	6	24	0	0	3	2.2481				8
35	37	37	6	37	0	0	3	2.2755				9
36	34	34	6	34	0	0	3	2.3302				10
37	25	25	6	25	0	0	3	2.4252				11
38	33	33	6	33	0	0	3	2.5502				12
39	36	36	6	36	0	0	3	2.5640				13

TABLE 36. TEST CASE 2 PROGRAM OUTPUTS, MODE 2 (CONCLUDED)

DATE 511 ID TESCAS2A  
TRADEOFF FOR  
PAIR 2 3

C8EFF		-2.60272		-39808		.45021		.62318		SCORE	0	1	2	3
	NO	ID	TY					CL						
1	39	39	6	39	0	0	3	-1.9086						1
2	38	38	6	38	0	0	3	-1.7356						2
3	16	16	4	16	0	0	1	-1.2687		1				
4	18	18	5	18	0	0	2	-.4892				1		
5	35	35	5	35	0	0	2	-.3849					2	
6	12	12	5	12	0	0	2	-.3162					3	
7	32	32	5	32	0	0	2	-.1598					4	
8	27	27	4	27	0	0	1	-.0390		2				
9	31	31	5	31	0	0	2	.1861					5	
10	37	37	6	37	0	0	3	.3591						3
11	26	26	5	26	0	0	2	.7572					6	
12	22	22	5	22	0	0	2	.8093					7	
13	17	17	4	17	0	0	1	1.1031		3				
14	20	20	6	20	0	0	3	1.1387						4
15	13	13	5	13	0	0	2	1.3804					8	
16	10	10	4	10	0	0	1	1.5012		4				
17	28	28	5	28	0	0	2	1.9348					9	
18	29	29	6	29	0	0	3	2.0391						5
19	30	30	6	30	0	0	3	2.7475						6
20	23	23	5	23	0	0	2	3.1291				10		
21	4	4	4	4	0	0	1	3.2665		5				
22	34	34	6	34	0	0	3	3.4228						7
23	24	24	6	24	0	0	3	3.6836						8
24	8	8	4	8	0	0	1	3.7523		6				
25	11	11	4	11	0	0	1	3.9418		7				
26	3	3	4	3	0	0	1	4.0626		8				
27	15	15	6	15	0	0	3	4.0817						9
28	5	5	4	5	0	0	1	4.2878		9				
29	7	7	5	7	0	0	2	4.7901				11		
30	6	6	4	6	0	0	1	4.8423		10				
31	9	9	5	9	0	0	2	5.1195				12		
32	21	21	6	21	0	0	3	5.2402						10
33	36	36	6	36	0	0	3	5.2593						11
34	14	14	5	14	0	0	2	5.3967				13		
35	2	2	4	2	0	0	1	5.4820		11				
36	1	1	4	1	0	0	1	5.4820		12				
37	19	19	5	19	0	0	2	5.8469				14		
38	25	25	6	25	0	0	3	7.2496						12
39	33	33	6	33	0	0	3	7.4226						13





TABLE 37. LOGICAL MODE OUTPUTS FOR TEST CASE 2 (CONTINUED)

32	33	5	3	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
----	----	---	---	----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--

TABLE 37. LOGICAL MODE OUTPUTS FOR TEST CASE 2 (CONCLUDED)

DATE 518 ID TESCAS2F  
LCOUNT = 6

	N0	ID	TY			CL	SCORE	0	1	2	3
1	17	17	4	17	0	0 1	00000000		1		
2	16	16	4	16	0	0 1	00000000		2		
3	12	12	5	12	0	0 2	00000000			1	
4	11	11	4	11	0	0 1	00000000		3		
5	10	10	4	10	0	0 1	00000000		4		
6	7	7	5	7	0	0 2	00000000			2	
7	6	6	4	6	0	0 1	00000000		5		
8	5	5	4	5	0	0 1	00000000		6		
9	4	4	4	4	0	0 1	00000000		7		
10	3	3	4	3	0	0 1	00000000		8		
11	2	2	4	2	0	0 1	00000000		9		
12	1	1	4	1	0	0 1	00000000	10			
13	18	18	5	18	0	0 2	00000020			3	
14	13	13	5	13	0	0 2	00000020			4	
15	22	22	5	22	0	0 2	00000021			5	
16	14	14	5	14	0	0 2	00000021			6	
17	9	9	5	9	0	0 2	00000021			7	
18	8	8	4	8	0	0 1	00000021	11			
19	27	27	4	27	0	0 1	00000060	12			
20	20	20	6	20	0	0 3	00000061				1
21	15	15	6	15	0	0 3	00000061				2
22	32	32	5	32	0	0 2	00000065			8	
23	31	31	5	31	0	0 2	00000065			9	
24	28	28	5	28	0	0 2	00000065		10		
25	26	26	5	26	0	0 2	00000065		11		
26	23	23	5	23	0	0 2	00000065		12		
27	35	35	5	35	0	0 2	00000067		13		
28	21	21	6	21	0	0 3	00000075				3
29	39	39	6	39	0	0 3	00000076				4
30	38	38	6	38	0	0 3	00000076				5
31	37	37	6	37	0	0 3	00000077				6
32	36	36	6	36	0	0 3	00000077				7
33	34	34	6	34	0	0 3	00000077				8
34	33	33	6	33	0	0 3	00000077				9
35	30	30	6	30	0	0 3	00000077				10
36	29	29	6	29	0	0 3	00000077				11
37	25	25	6	25	0	0 3	00000077				12
38	24	24	6	24	0	0 3	00000077				13
39	19	19	5	19	0	0 2	00000077		14		

## THE KARNAUGH MAP METHOD

Karnaugh mapping is a graphic procedure which assigns a map region to each logical variable. In one half of this region the logical variable is true, in the other half it is false. Within each half, additional regions may be defined for additional variables, which themselves have a true half and a false half. The two-variable example in Figure 17a will illustrate this. In the top half of the map, variable B is true, while in the bottom half it is false. Variable A is true on the left half of the map, and false on the right half. Four "states" are thus defined: (A true, B true), (A true, B false), (A false, B true), (A false, B false). States may be combined as follows: to combine the two states (A true, B false) and (A false, B false) variable A may be ignored; the combination of the above two states may be called (B false), since it is defined for any A (true or false) when B is false. Note that the two states to be combined share a boundary. To combine two states, either A or B should be the same in both states. Thus, these two states [(A true, B false) and (A false, B true)] may not be combined, for this map. (Note that these two states do not share any boundaries.) A Karnaugh map of three logical variables is shown in Figure 17b. There are eight states in this map; if true is represented by the value one and false by value zero, the eight possible states are:

<u>A</u>	<u>B</u>	<u>C</u>
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Note that each logical variable is true in half the states, and false in the other half. On the actual Karnaugh map, the false states of variable C are split to

prevent C from always being the same as A, or always the opposite of A, since this would be the same as a two-variable situation. States may be combined as with two variables; combining states (A true, B false, C false) and (A false, B false, C false) gives the result (B false, C false) for any A.

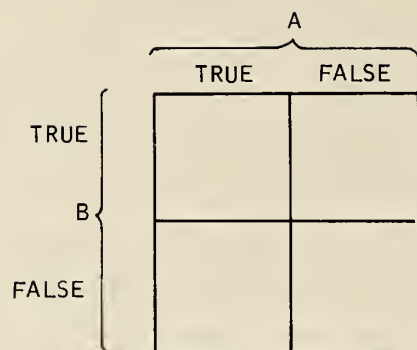
Note that this combined state is possible because of the shared boundary of the two (C false) regions; that is, the map may be thought of as being continuous by joining the ends or the top and bottom. The boundary at the left edge of the map is the same boundary as that at the right edge, and the boundary at the top is the same boundary as at the bottom.

Combined states may be further combined; again the requirement is for a common boundary or unchanging variable. For example, state (A true, B true, C false) and state (A true, B false, C false) may be combined to (A true, C false). Similarly, state (A true, B true, C true) and state (A true, B false, C true) combine to form state (A true, C true). Now the two combined states may be joined to form the state (A true) for all B and C. (Note this state (A true), with only one variable defined, comprises half of the Karnaugh map of Figure 17b.)

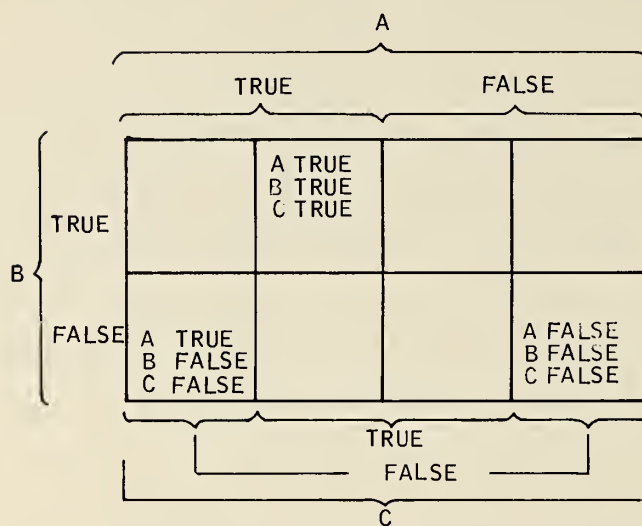
The utility of combining states comes from reducing the number of variables which must be specified to determine the desired state. In the classifier program, the result of the logical method using six variables was a list of 10 states (out of a possible 64), with the three classes spread through them. Combining states with Karnaugh mapping techniques may allow specifying a single class made up of several states without using all six variables. The Karnaugh map of the method three (logical mode) results is shown in Figure 18.

Many of the states on the map have no feature vectors in them, since only ten distinct groups were formed by the logical mode classifier. These empty states may be used to produce combined states, since they contain no values, and might produce a simpler expression by eliminating one or more of the

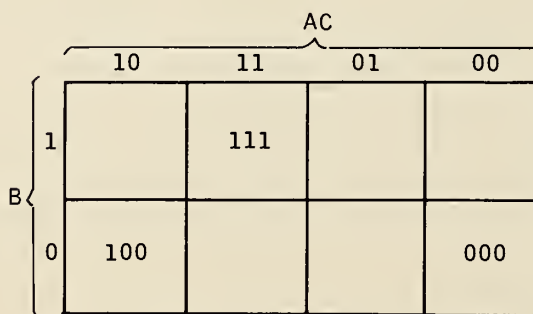




(A)



(B)



(C)

ALTERNATIVE  
REPRESENTATION  
OF (B)

Figure 17. Karnaugh Map Example



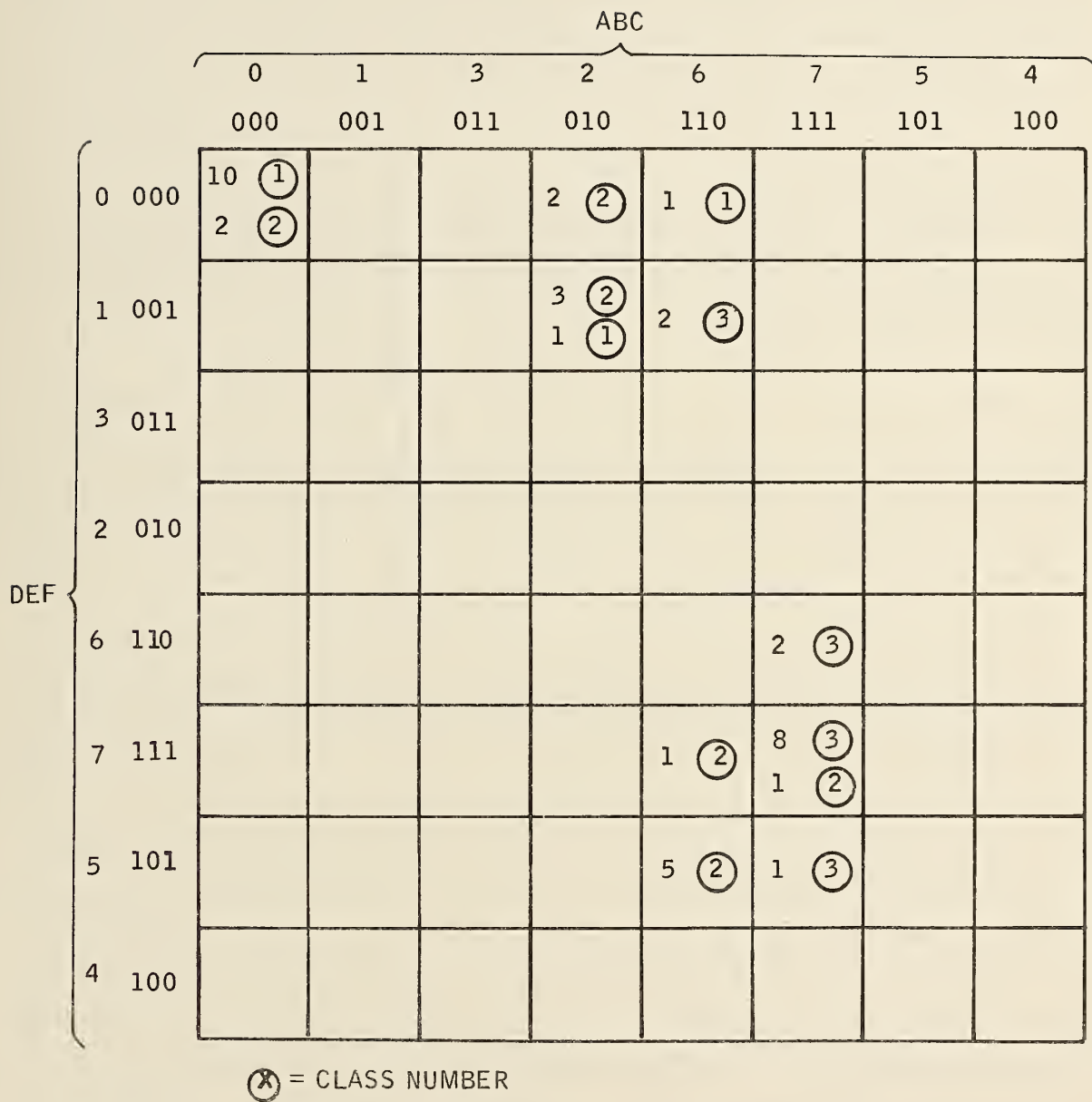


Figure 18. Six Variable Karnaugh Map

six variables. For example, any feature vector which has all of its variables false will be in the region of the map (000000). However, since the adjacent column is empty, the variable C may be ignored; the region then may be defined as (00X000), where X means a "don't care" variable. This combining does not introduce any undesired feature vectors, nor does it leave out any desired vectors. Further, variables d, e, and f may all be ignored, thus the combined region is (A false, B false), or 00XXXX (X = don't care), for any c, d, e, f values. Further combining to determine presence of this state is B; any feature vector which has (B false) is in this region, regardless of the states of any other variables.

The combinations described here are not the only ones that are available. Different combinations may be formed, as required, to include different states of the Karnaugh map. These states are identified in terms of the six logical variables by assigning value one to a true variable and value zero to a false variable. Thus, the state (A true, B true, C true, D true, E true, F false) may be represented as state 111110. This is one state of the possible 64 in a six-variable Karnaugh map. State (A true, B false, C false, D true, E false, F true) would be represented as 100101. By using the Karnaugh map, the original three classes may be defined in terms of the states which contain vectors of each class, and, because of the many empty states ("don't care" states which neither contribute nor eliminate feature vectors), the expression for a given class will probably not need to include all six of the logical variables. In the Karnaugh map for the logical mode classification of test case two, the classes may be identified by variable values as follows: class 1 is  $\left[ (B \text{ false}) \text{ or } (A \text{ true, E false, F false}) \right]$ , or equivalently, any state X0XXXX or 1XXXX00 is a member of the group of states comprising class 1; class 2 is all states 01XXXX or XX01XX; class 3 is XX1XXX or 1XX0X1. Although each of the six variables is used at least once in all these terms, no term uses all six, therefore class determination is simplified in terms of the number of variables needing to be specified. This grouping includes four misclassified vectors; this misclassification occurs when a single state contains

vectors from more than one class, such as state 000000 which contains mostly class 1 feature vectors. If state 000000 is called a class 1 state, the two vectors in state 000000 from class 2 will be misclassified; state 010001 contains vectors from two different classes, as does state 111111. This mixing of classes within a single state results from the selection of pairwise boundaries, and is analogous to misclassified feature vectors found in method 1 or method 2 executions.

A simplification of the class determination from the logical mode results may be obtained at the expense of additional misclassified feature vectors. For test case 2, these additionally misclassified vectors are those in states 110,000 and 110,001. If these two states are made part of class 2, the determination of classes becomes:

Class 1 for all (B false), states X0XXXX

Class 2 for all (B true, C false), states X10XXX

Class 3 for all (A true, C true), states 1XXXX.

(Note that variables D, E, and F may be ignored completely in this determination, however, there are now seven misclassified feature vectors instead of four.)

The data set was projected onto a subspace by using a "minus eight" card along with a pair of "two" cards. The second field of the "minus eight" card was punched with the value 1 for the subspace projection; the subspace to be used was defined using the "two" cards. Since, in three dimensions, a plane is defined by any three points, the spanning set of vectors to define the plane was chosen as two of the vectors representing the differences between the class means. One of these vectors was found by subtracting each component of the mean vector of class 1 from the corresponding component of the mean vector of class 3; the other vector was similarly found by subtracting each component of the mean vector of class 2 from the mean vector of class 3, as shown here:

- Class 1 mean vector:           6.417 in feature 1 direction  
                                  6.083 in feature 2 direction  
                                  8.750 in feature 3 direction
  
- Class 2 mean vector:           10.070 in feature 1 direction  
                                  9.143 in feature 2 direction  
                                  7.214 in feature 3 direction
  
- Class 3 mean vector:           12.620 in feature 1 direction  
                                  10.080 in feature 2 direction  
                                  9.769 in feature 3 direction
  
- Spanning vector A (3-2):       2.550 in feature 1 direction  
                                  0.937 in feature 2 direction  
                                  2.555 in feature 3 direction
  
- Spanning vector B (3-1):       6.203 in feature 1 direction  
                                  3.997 in feature 2 direction  
                                  1.019 in feature 3 direction

These spanning vectors are input to the program using coefficient cards ("two" cards), one for each spanning vector; the second and fourth fields of these "two" cards are not used by the program and may contain any value, however, the scaling factor (third field) is required. The data set is then projected onto the plane defined by the two spanning vectors (two vectors having a common origin define a plane), or onto its orthogonal complement (a line for this case), depending upon the value in the second field of the "minus eight" card. This projection produces new feature values for each feature vector; the program then executes using this re-defined feature vector set.

A subspace of a three-dimensional space may be either a two-dimensional space (plane), or a one-dimensional space (line). The orthogonal complement (in the three-dimensional case) of the two-dimensional subspace (plane)



is the normal line to that plane, and conversely. For a four-dimensional space, a subspace could be of three dimensions, or of two dimensions, or of one dimension; the orthogonal complement to these would be of one, two, or three dimensions, respectively. By projecting a data set onto a subspace, all variations in the direction orthogonal to the subspace is eliminated, simplifying the data representation and possibly permitting a simpler solution to the classification problem.

Table 38 shows the results of projecting the test case 2 data onto the plane of the means of the three classes. Since this plane is in a three-dimensional space, three coordinates are needed to specify any point, and that is the reason for there being three coordinates for points in the two-dimensional space.

To plot this data on a two-dimensional surface, any two of the projected coordinates can be used. Figure 19 shows the data plotted using projected features 1 and 2. This plot can be compared with Table 27 to see the effect of the third feature upon the data distribution. (The points in both figures have been connected together in the same way by lines. Comparing the shapes thus outlined will give an idea of how the third feature warped the distributions.)



TABLE 38. TEST CASE 2 DATA, PROJECTED\* NUMERICAL VALUES

Index	Feature 1	Feature 2	Feature 3	Index	Feature 1	Feature 2	Feature 3
1	5.568	0.324	10.784	21	12.140	3.368	15.460
2	6.809	0.410	13.143	22	12.443	6.934	5.317
3	6.813	1.405	10.142	23	13.314	6.687	7.904
4	6.961	2.193	8.072	24	14.774	8.597	5.213
5	6.411	1.549	8.859	25	15.935	7.936	9.664
6	7.871	3.459	6.167	26	12.552	6.210	7.739
7	7.980	2.734	8.589	27	12.700	6.998	5.669
8	9.736	6.220	1.758	28	14.157	7.913	5.979
9	9.842	4.501	7.181	29	15.180	9.449	3.495
10	8.790	4.437	5.152	30	12.802	4.283	14.094
11	9.478	2.884	11.300	31	13.539	7.228	6.745
12	9.488	5.869	2.295	32	14.125	8.390	4.468
13	10.215	5.829	3.951	33	16.883	7.442	13.162
14	11.813	5.542	8.195	34	14.079	4.887	14.963
15	12.839	8.073	2.709	35	14.527	8.246	5.751
16	8.723	4.397	5.131	36	18.166	10.036	8.028
17	10.726	4.961	7.656	37	14.488	6.734	10.243
18	11.022	6.537	3.516	38	14.455	7.211	8.731
19	13.491	6.003	10.347	39	14.748	7.792	7.593
20	13.064	8.614	1.549				

\*Projection upon the plane of the means

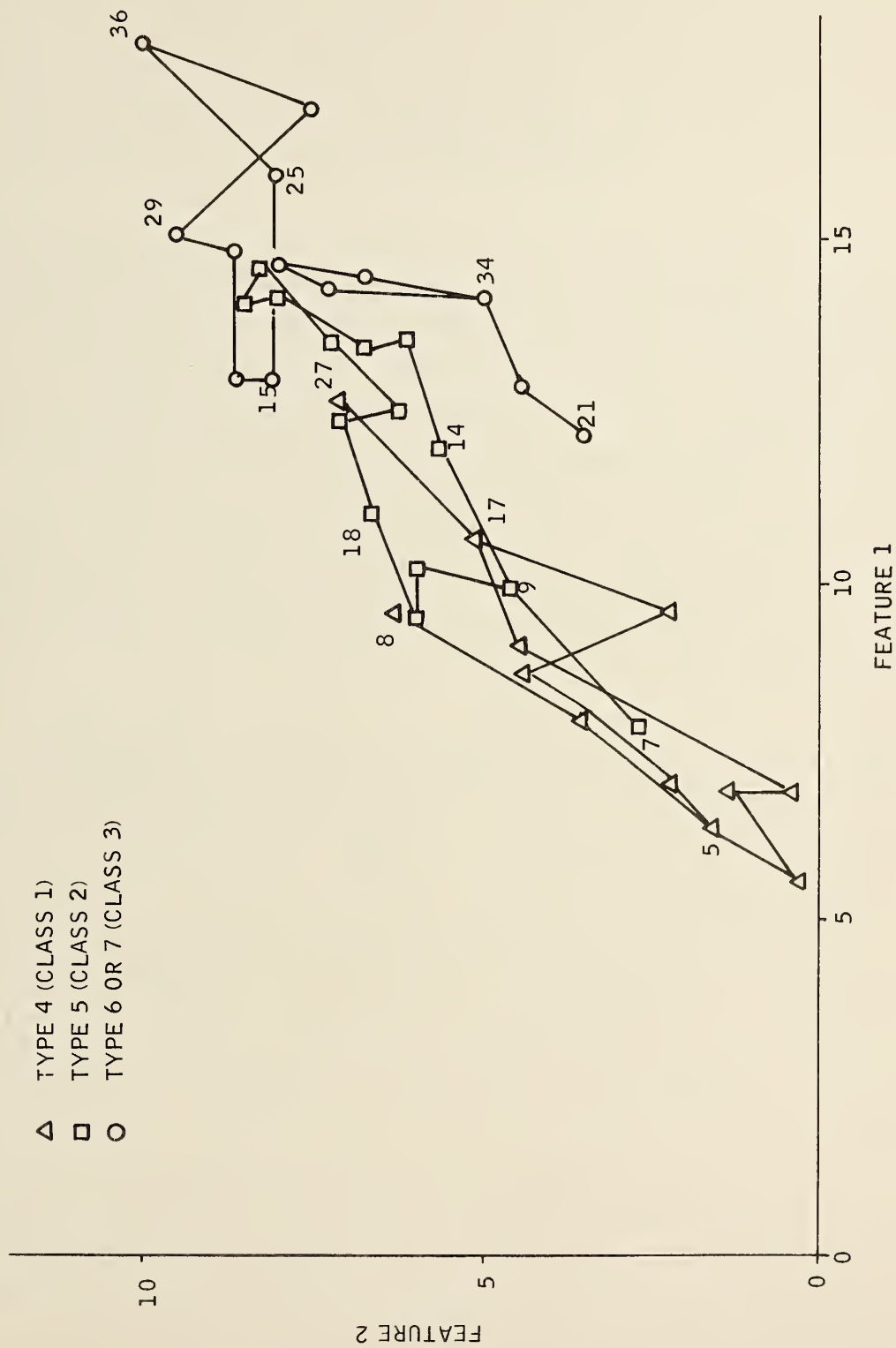


Figure 19. Plot of Test Case Two Data Projected Onto the Plane of the Means

## CHAPTER 5

### PROGRAM STRUCTURE

The program is comprised of a main program block and 10 subroutines; five additional library subroutines are called during program execution. The main program block performs the functions of data and control card input, data preparation, statistical calculation, classification, and output listing. Eight of the subroutines perform matrix and vector arithmetic such as dot products, cross products, addition, multiplication by a scalar; one subroutine evaluates eigenvalues; and one is used to calculate and print tradeoff data based on feature values or feature vector scores. One of the library subroutines performs matrix inversion; the others used by the program file manipulations such as reading and writing.

The main program functions are structured as modules of code within the main program, and are jumped to by the main control loop. When one of these modules is "entered" by the program, it performs its function based on status flags which are set or cleared by other modules. When the function of a particular module is complete, a jump is made back to the main control loop. This main control loop is an infinite loop, and contains programmed pauses to allow the operator to input data or otherwise control program execution. Use of this infinite loop is permitted by the system monitor, since control is returned to the monitor when an execution is complete and no data remains to be input.

There are three programmed pauses in the program; each has a number which is printed on the typewriter input/output device whenever the pause is encountered by the program. The program also tests two console switches, called sense switch one and sense switch six. If sense switch one is on, the program will enable the typewriter for data input or modification prior to

execution; if sense switch six is on, the program will suppress the feature vector listing on the output. Sense switches should be set before starting the program run.

Programmed pause number 1 occurs upon completion of an execution, and allows the operator to change magnetic tapes, set switches, or load the next set of cards into the card reader before proceeding. To continue the run, a carriage return should be typed by the operator. Pause 2 occurs when the program detects an error in the input cards; if either the number of classes (field two of "three" card), number of types (field two of "four" card), or number of features (field two of "five" card) is zero, the program will execute a pause 2. At that time, the operator may continue the run by typing a carriage return, in which case errors in output will result; or sense switch 1 may be turned on, followed by a carriage return, in which case the program will accept new or revised data from the typewriter input device. This new or revised data must follow a specified format to ensure program recognition. An example of the required format is: "NC = 3". This means "the variable named NC should be set to the value 3". (The quote marks are not part of the format; they are used here to delimit the format required characters.) Following a statement such as "NC = 3," a carriage return is typed, and another data value may be input, such as "NT = 6," followed by a carriage return, followed by as many new or revised data values as desired, in the specified format, each followed by a carriage return. To terminate this input operation, type an upper case Q, then carriage return. Pause 3 occurs when the program detects that no samples (feature vectors) have been selected for classification. This would occur if no class were specified for any of the feature vector types being used in the current execution (see "four" card discussion).



## FIELD DELIMITERS

Most cards used by the program have a format with the first field specified as three card columns wide and other fields on the card being defined as "widthless" or free format. This means the fields are set off from each other by certain characters, called delimiters, rather than by being a certain number of columns in width. Comma characters and blank characters (space characters) are commonly used as delimiters. Any comma is read as a delimiter, but the only spaces (blanks) which are read as delimiters are those which are preceded by a digit (0-9) or decimal point. Other blanks are ignored, and empty fields are read as zeros.



TABLE 39. TABLE OF VARIABLES AND ARRAYS  
ACCESSIBLE TO THE PROGRAM WRITER  
THROUGH THE USE OF CONTROL CARDS  
AND DATA CARDS

Name	Use Immediately Prior to Execute (-99)
CLASS (I)	Class to which type I is assigned
FEAT (1:20, I)	Feature vector data for $I^{\text{th}}$ feature vector sample: FEAT (1, I) = I FEAT (2, I) = sample i. d. number FEAT (3, I) = sample type FEAT (4, I) = number of features in sample FEAT (5, I) FEAT (7, I) = 0 to suppress } in the case } $\neq 0$ to transform } MODE 2 $\neq 0$ FEAT (11:20, I) = feature vector
FLAG (1)	Number of feature vector cards read since last execute
FLAG (3)	Number of coefficient vector cards since last execute
FLAG (4)	Number of feature tradeoff listings (= number of -7 cards since last execute)
ITV (I)	Feature # to be defined by the $I^{\text{th}}$ coefficient vector
ITV (10 + I)	Scaling factor for feature defined by the $I^{\text{th}}$ coefficient vector
KMODE	KMODE = 1 for method 1 classifier KMODE = 2 for method 2 classifier KMODE = 3 for method 3 classifier (logical mode) KMODE = 0 for no classification (default value)
LCOUNT	Number of features to be used by logical mode
LIST (I)	LIST (I) $\neq 0$ for $I^{\text{th}}$ list option. Only options 5, 6, and 7 implemented
MODE 2	MODE 2 = 1 for secondary feature transformation 1 MODE 2 = 2 for secondary feature transformation 2 MODE 0 = for no secondary feature transformations (default value)
NC	Number of Classes
NF	Number of Features to be used during execution
NFV	Number of Feature Vectors to be used in execution
NPCON	NPCON = 1 for projection to subspace NPCON = 2 for projection to complement NPCON = 0 for no projection (default value)
NT	Number of Types
PROB (I)	Probability for class I
SFEAT (I)	$I^{\text{th}}$ selected feature
TM1 (1:10, I)	$I^{\text{th}}$ coefficient vector
TV1 (I)	Constant for feature defined by the $I^{\text{th}}$ coefficient vector

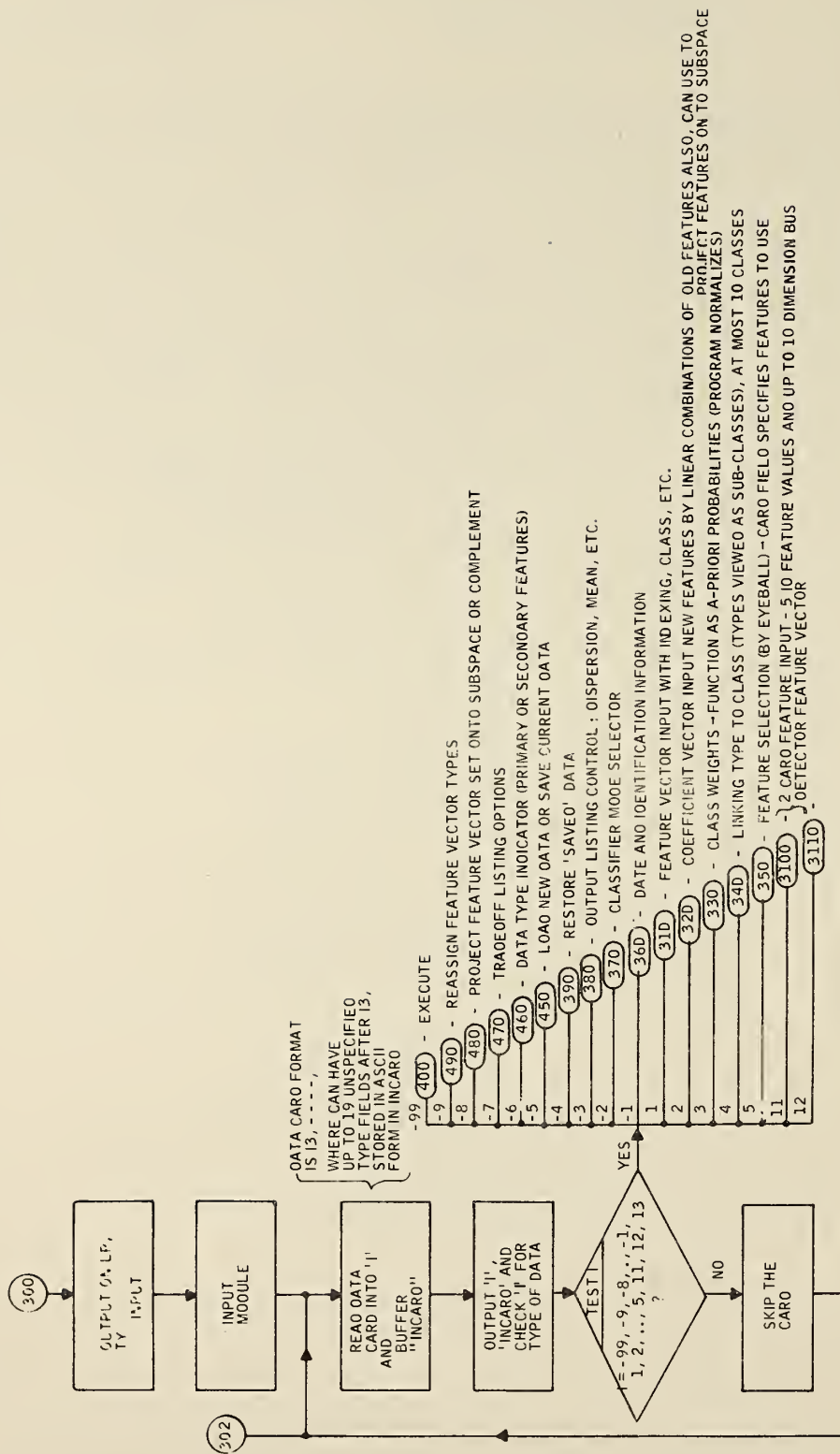


Figure 20. Flow Charts

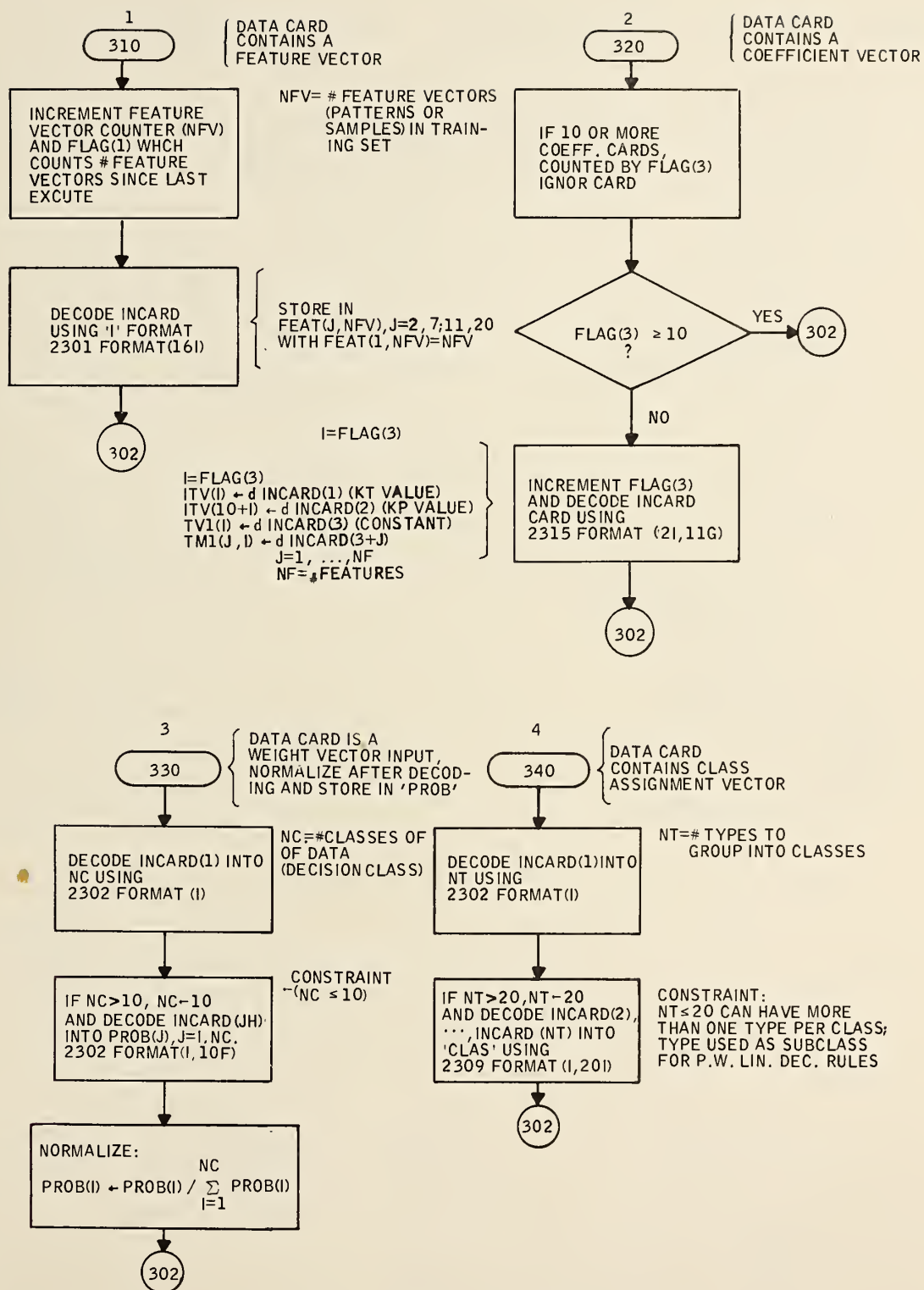


Figure 20. Flow Charts (Continued)

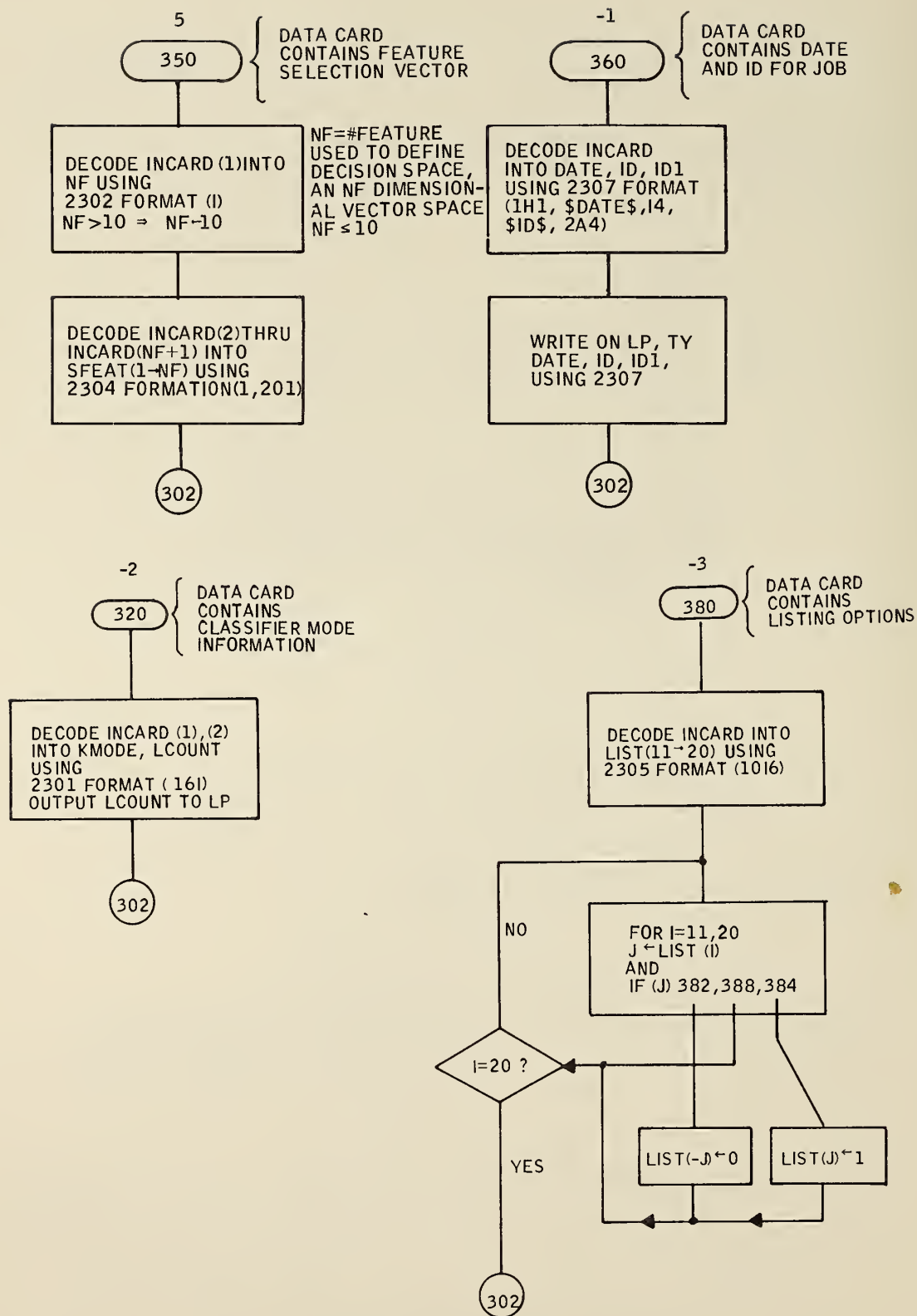


Figure 20. Flow Charts (Continued)

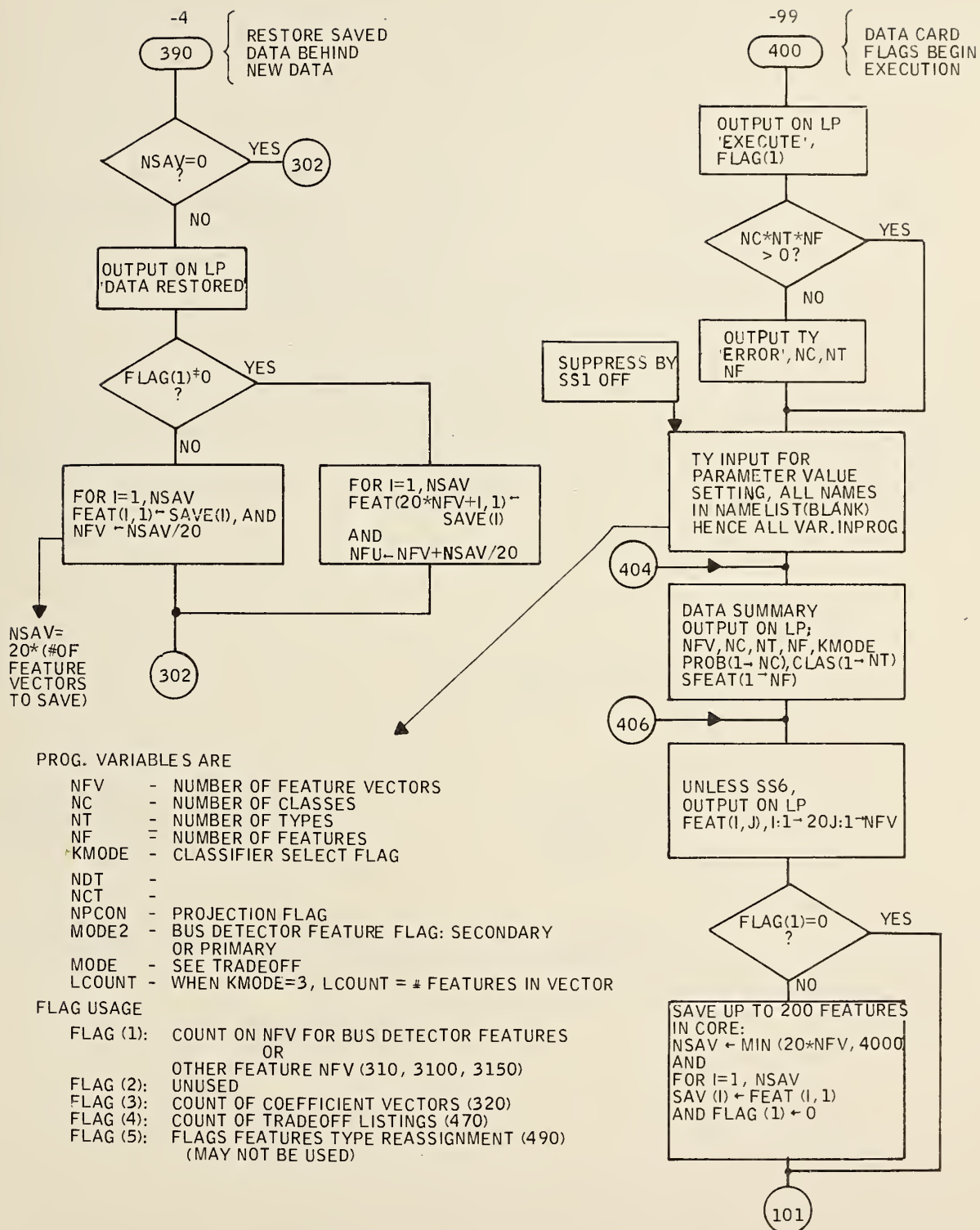


Figure 20. Flow Charts (Continued)



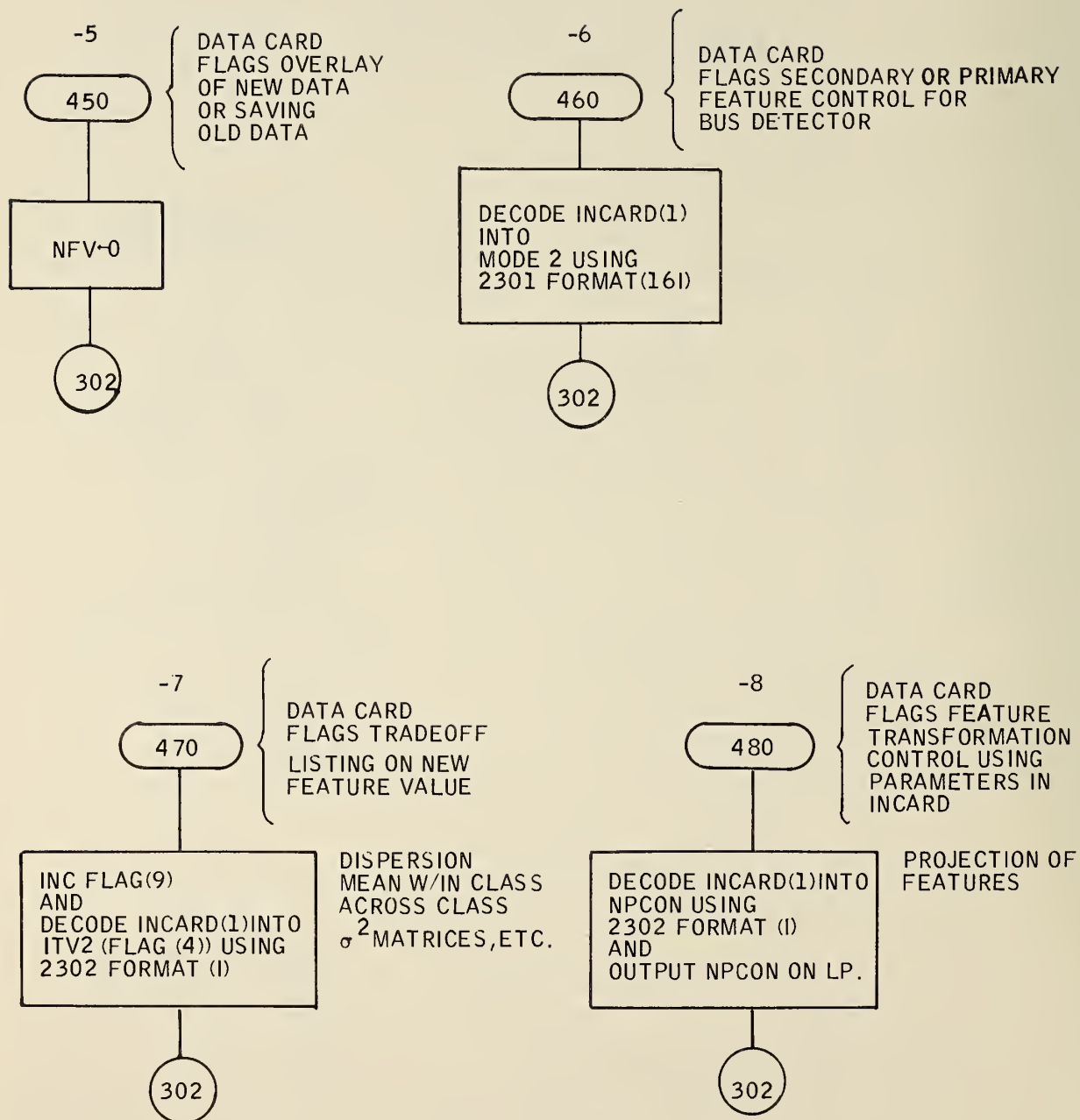


Figure 20. Flow Charts (Continued)

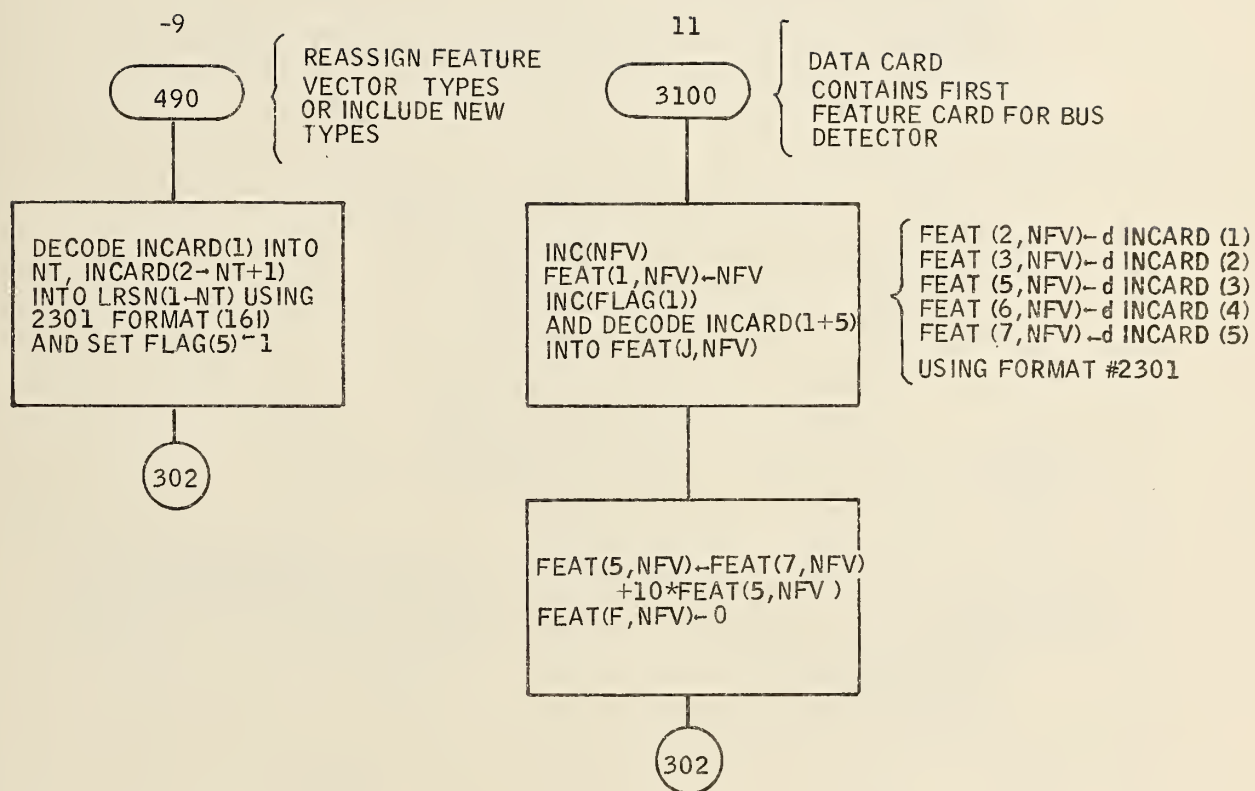


Figure 20. Flow Charts (Continued)

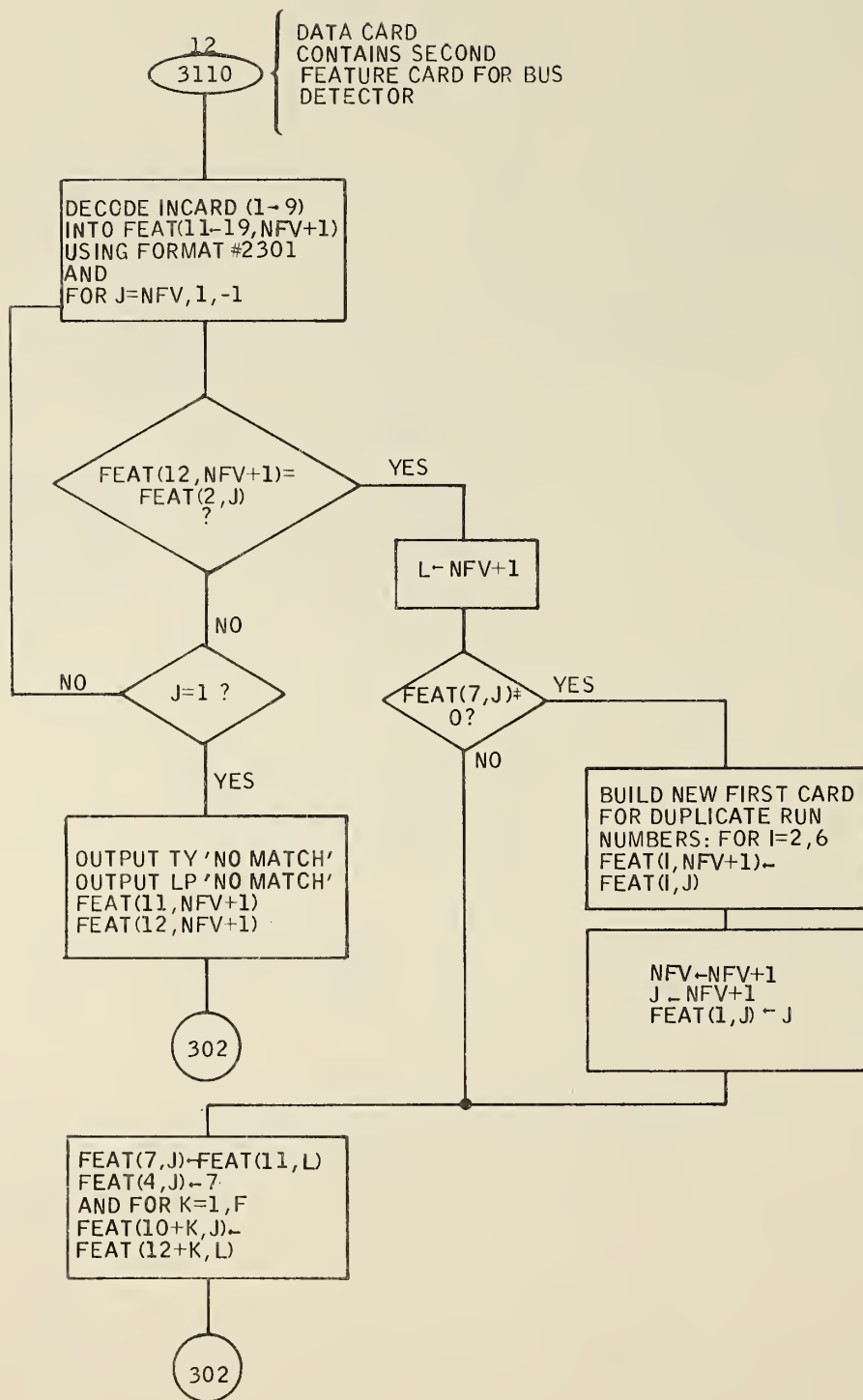


Figure 20. Flow Charts (Continued)

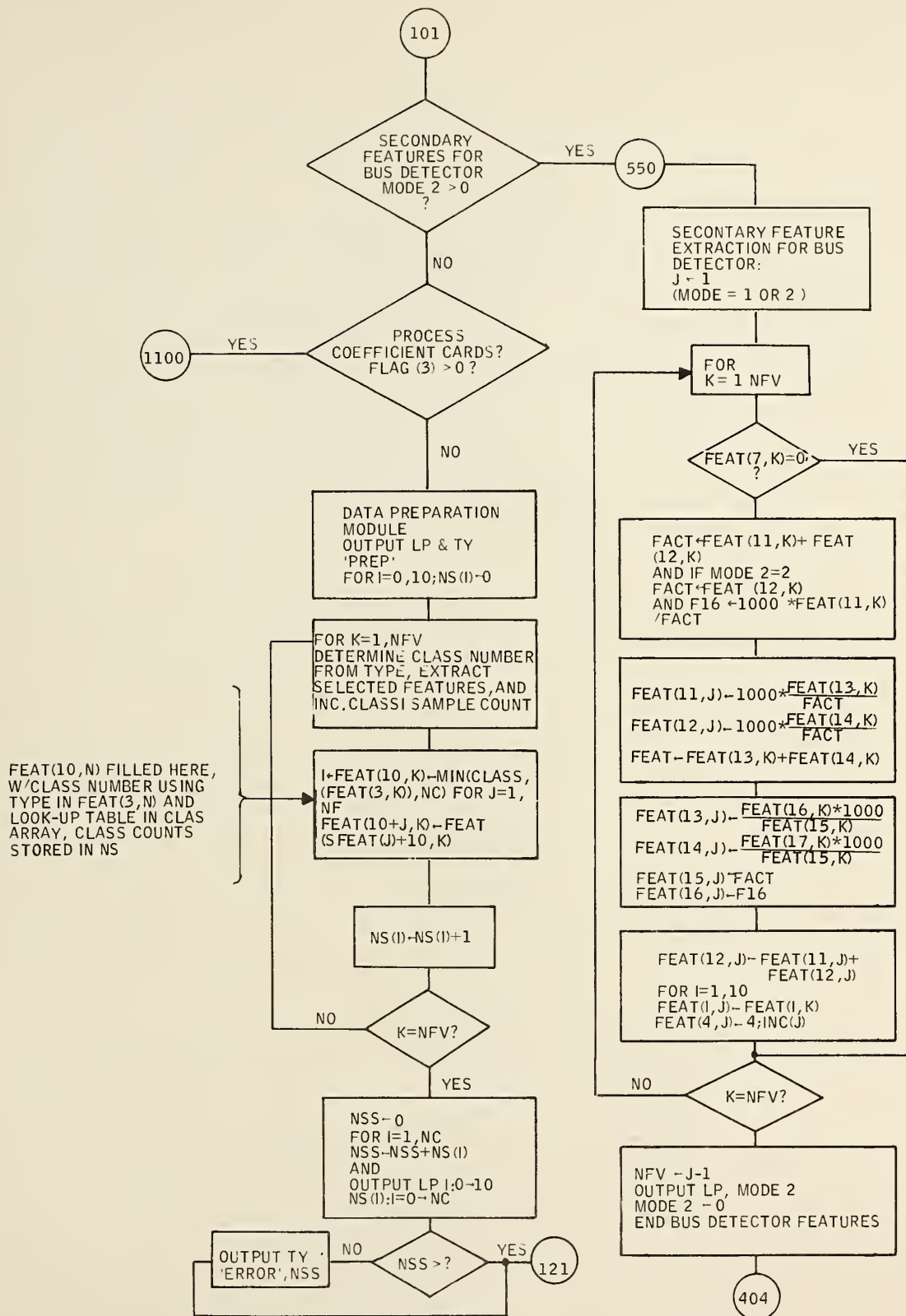


Figure 20. Flow Charts (Continued)

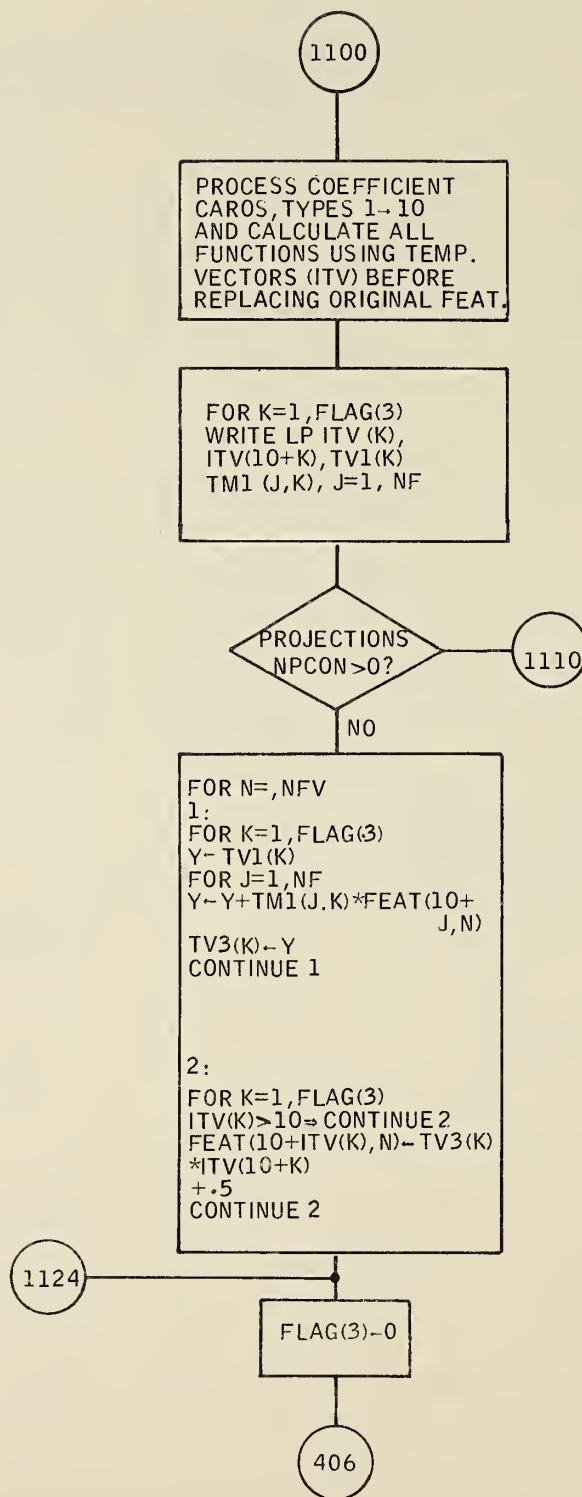


Figure 20. Flow Charts (Continued)



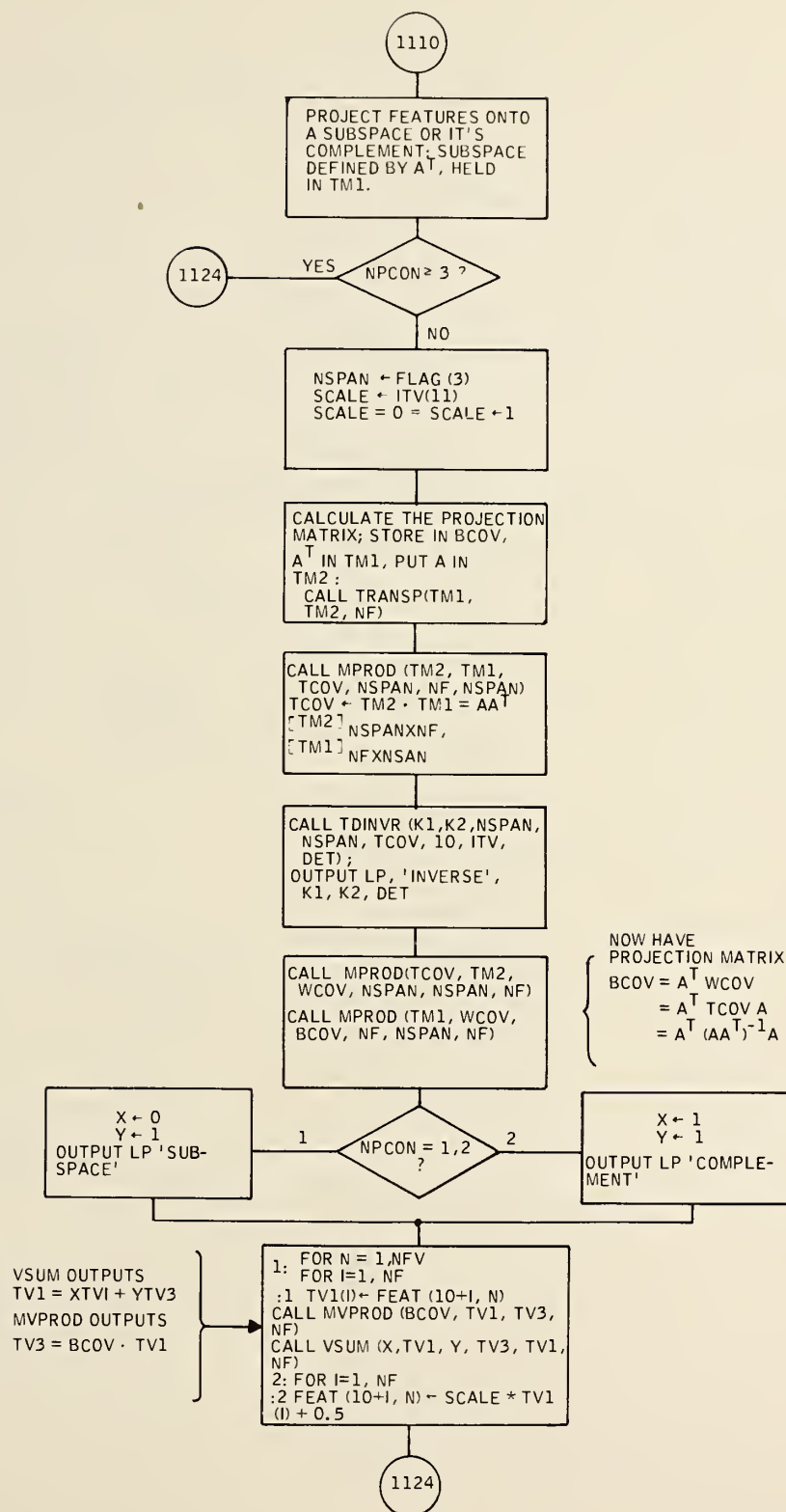


Figure 20. Flow Charts (Continued)

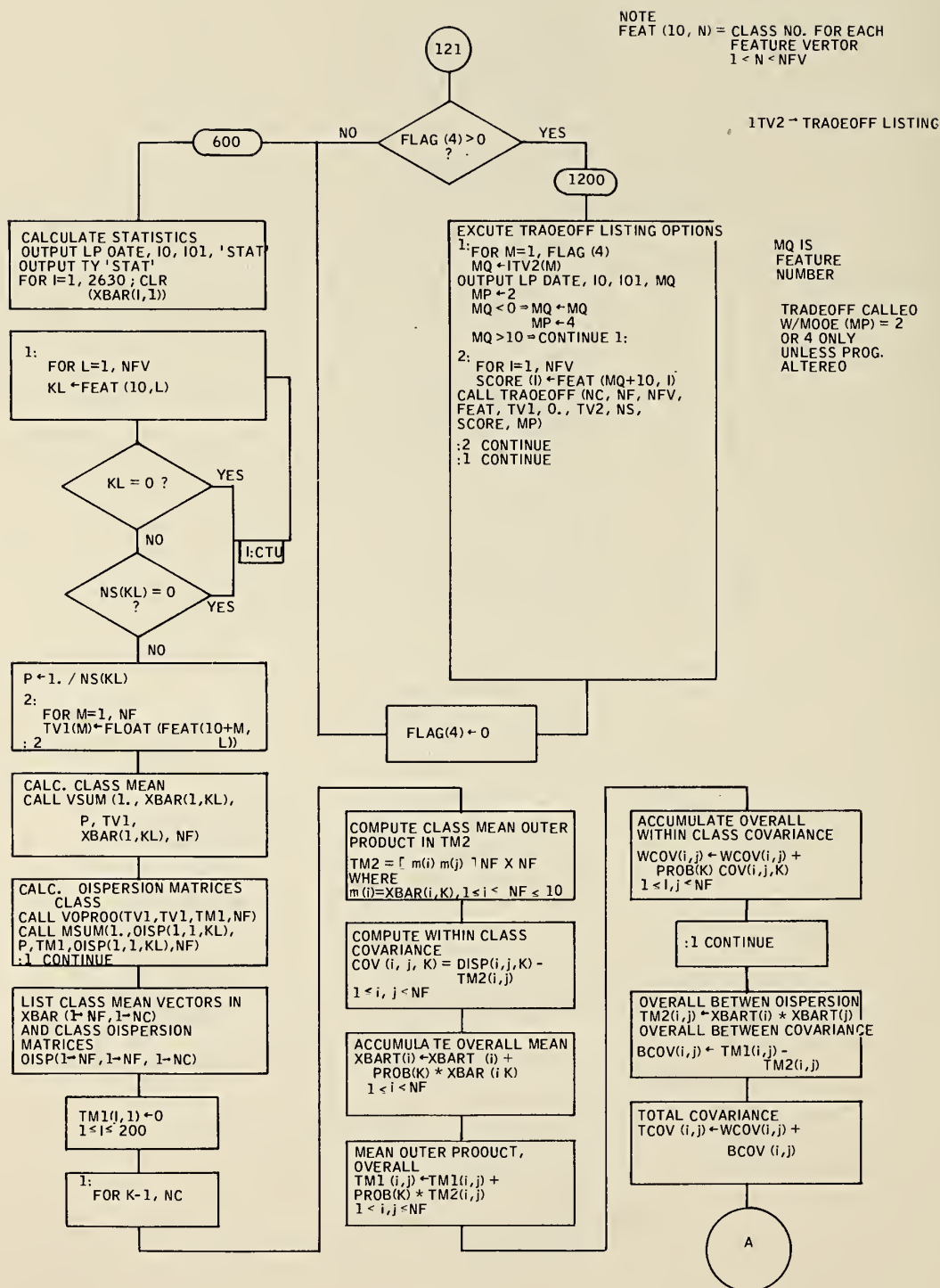


Figure 20. Flow Charts (Continued)



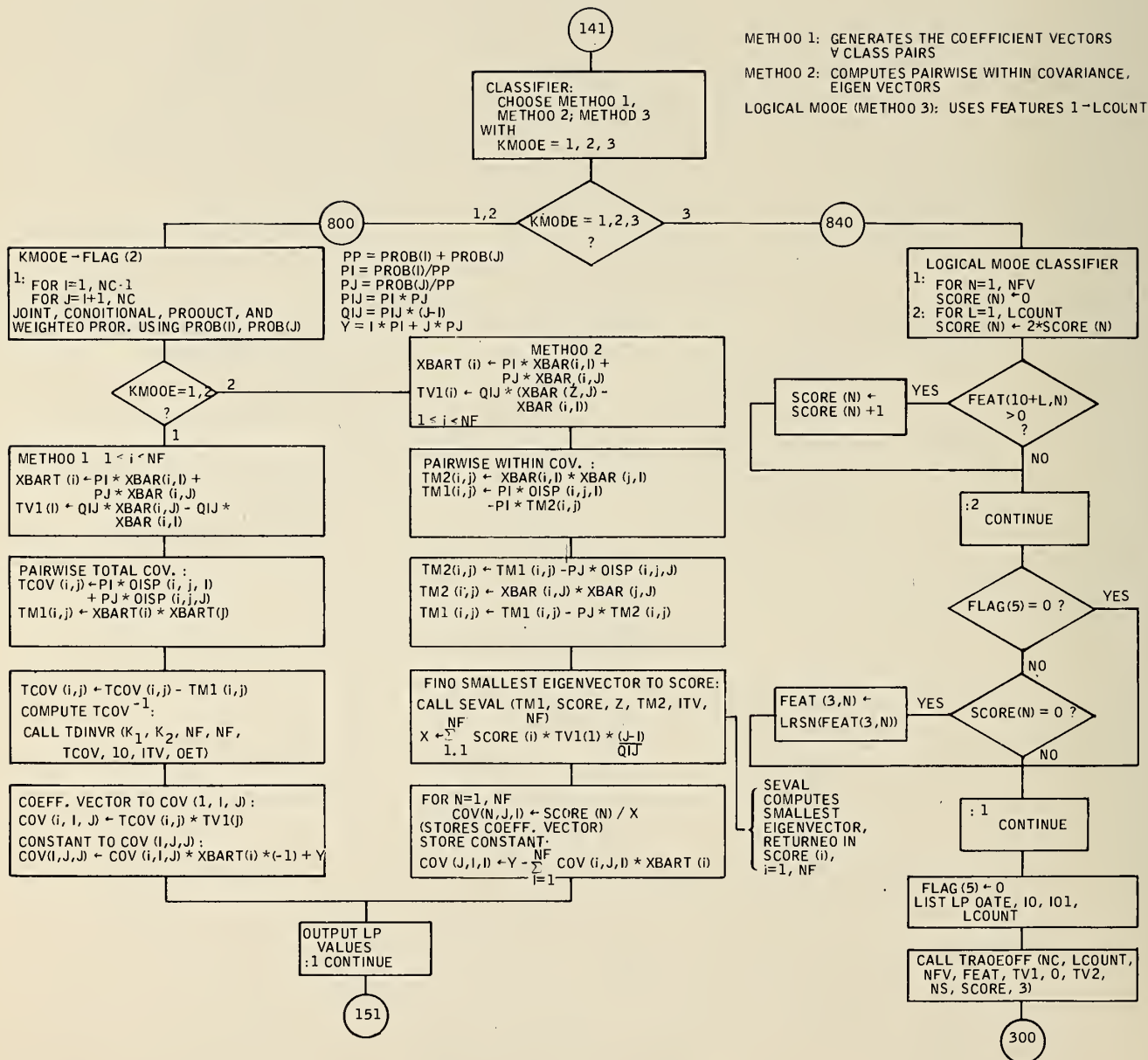


Figure 20. Flow Charts (Continued)

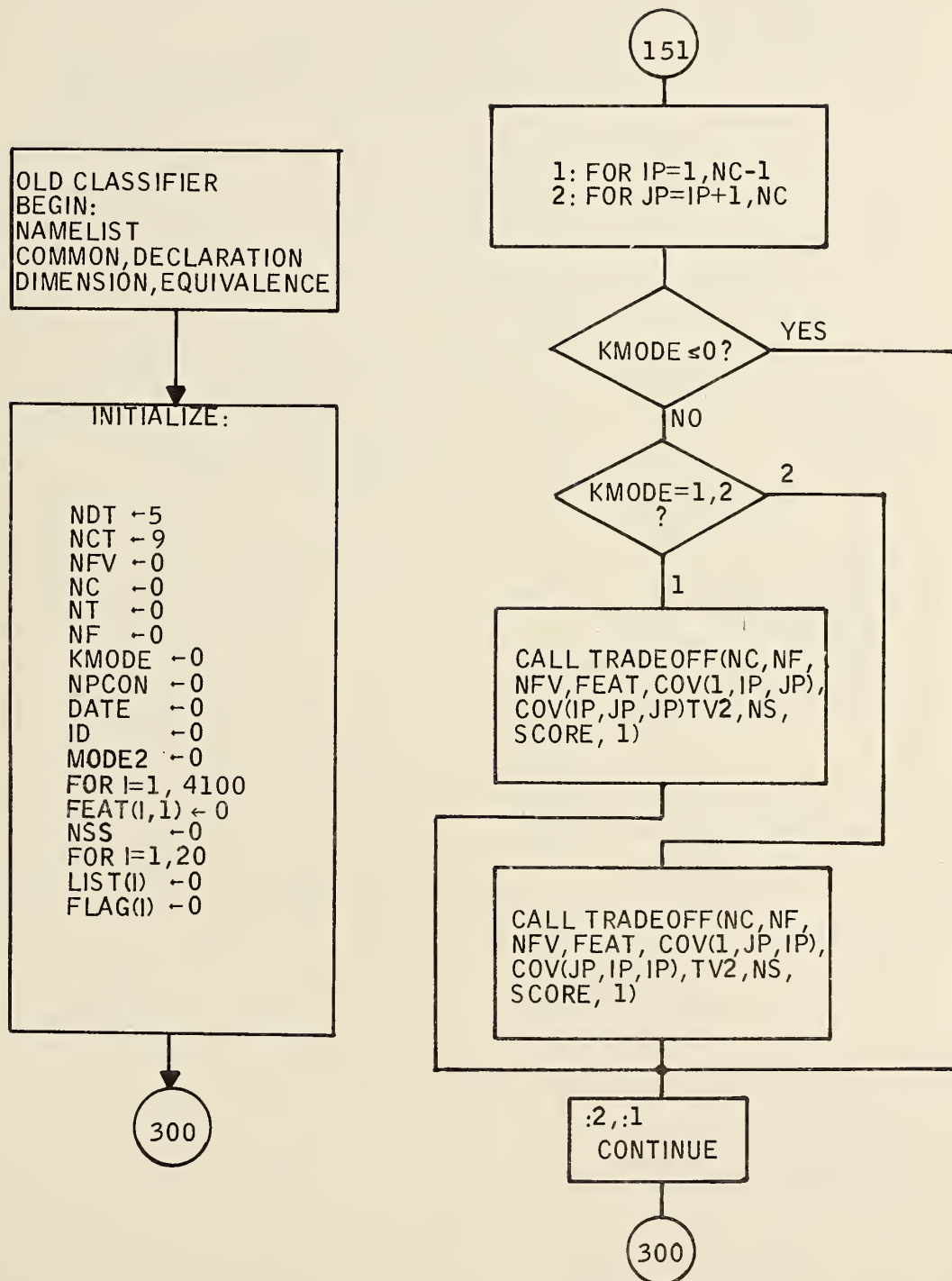


Figure 20. Flow Charts (Continued)



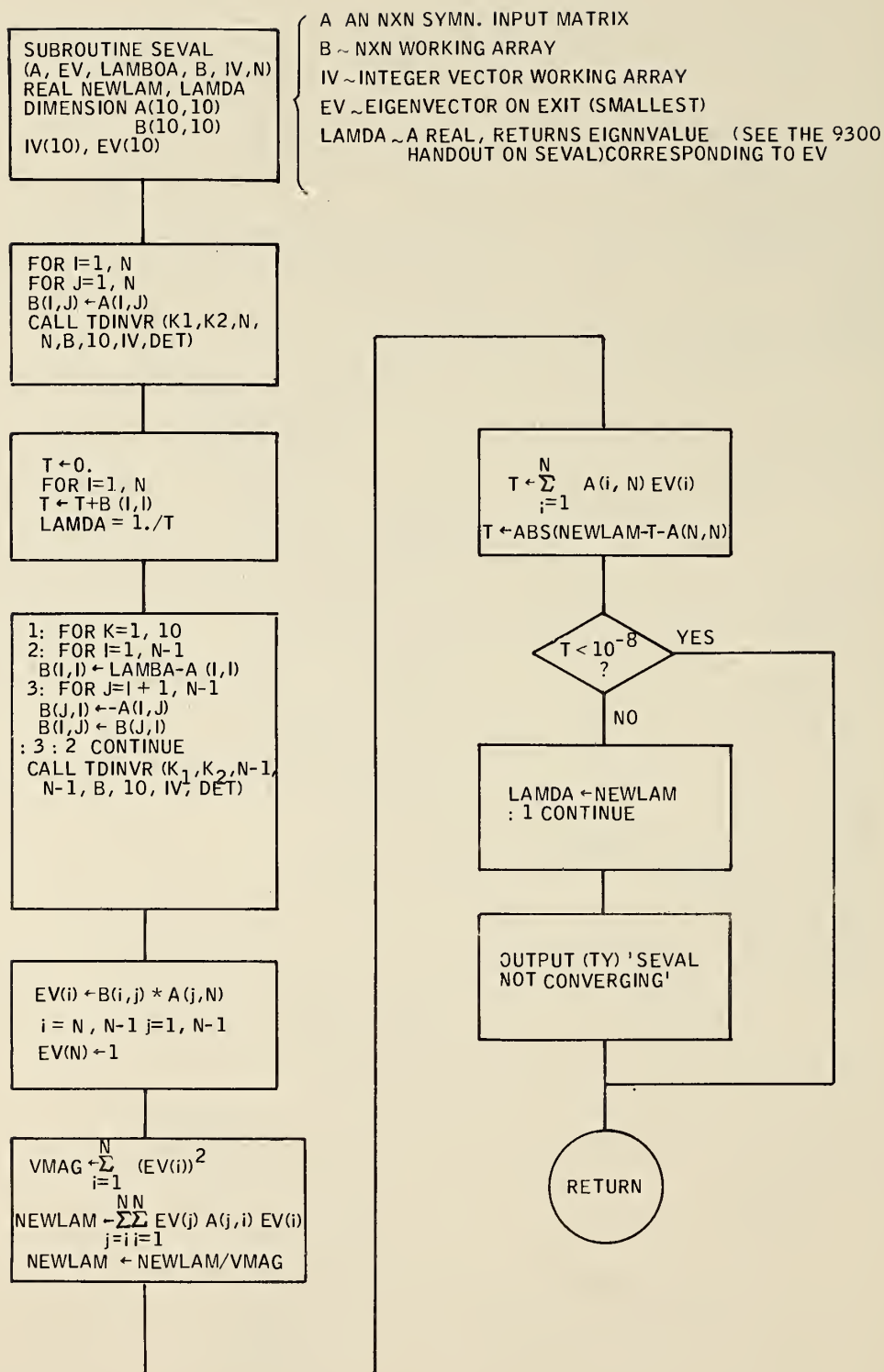


Figure 20. Flow Charts (Continued)

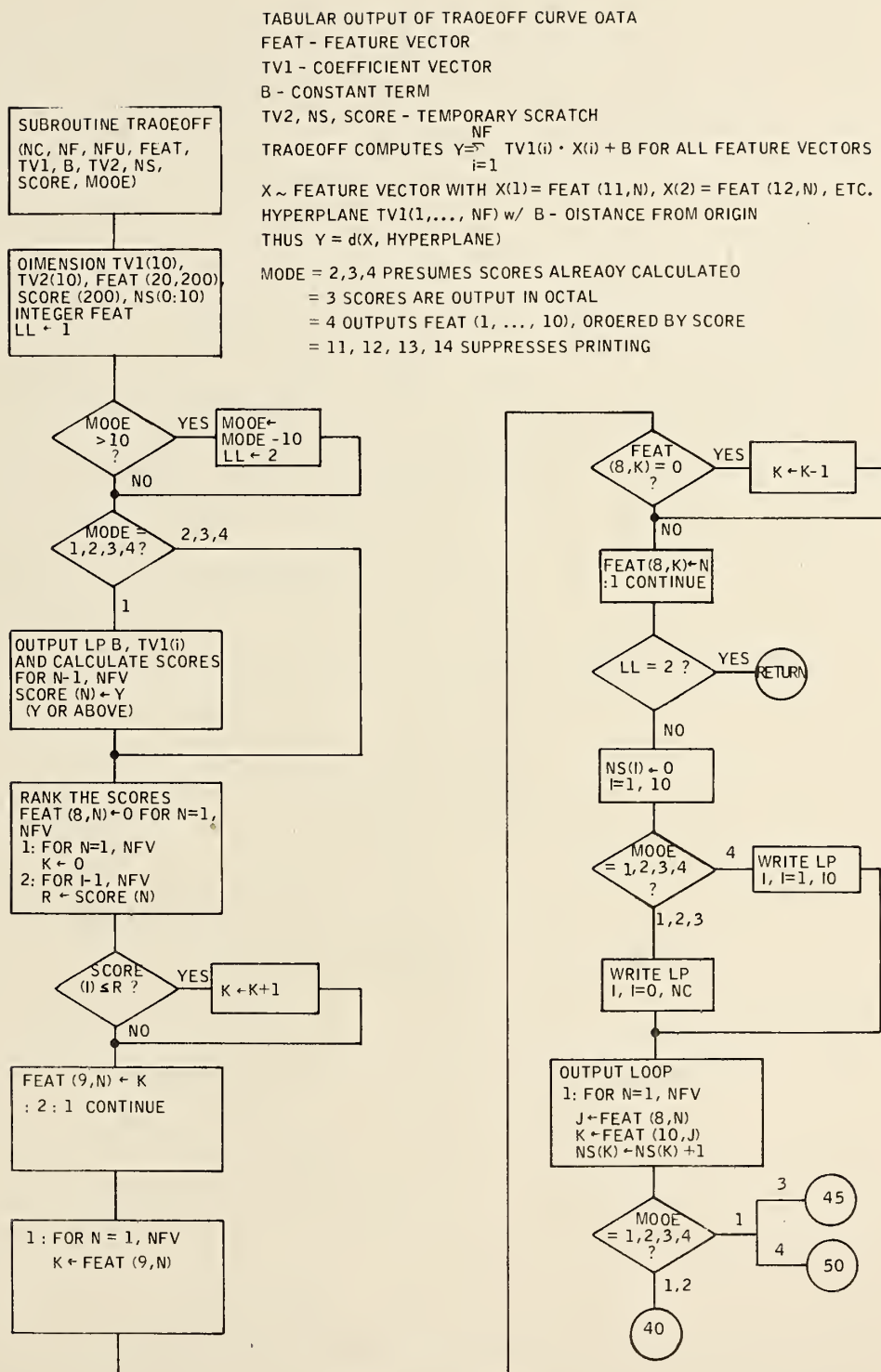


Figure 20. Flow Charts (Continued)

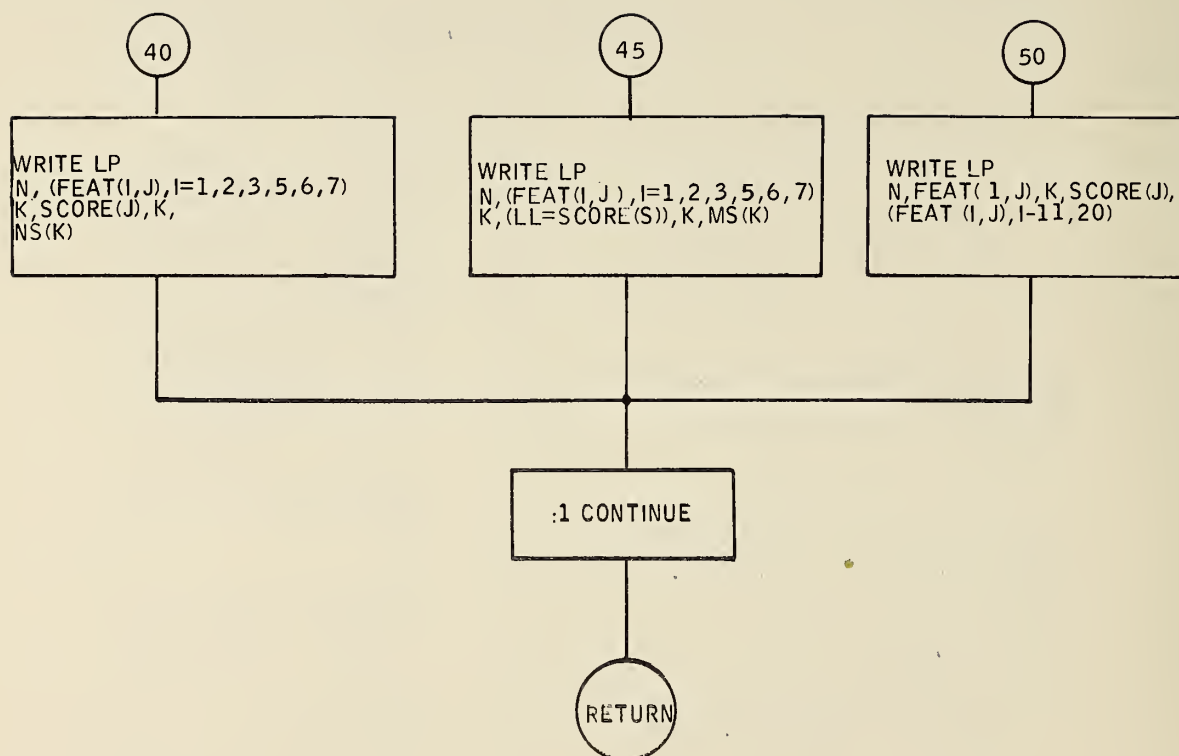


Figure 20. Flow Charts (Continued)

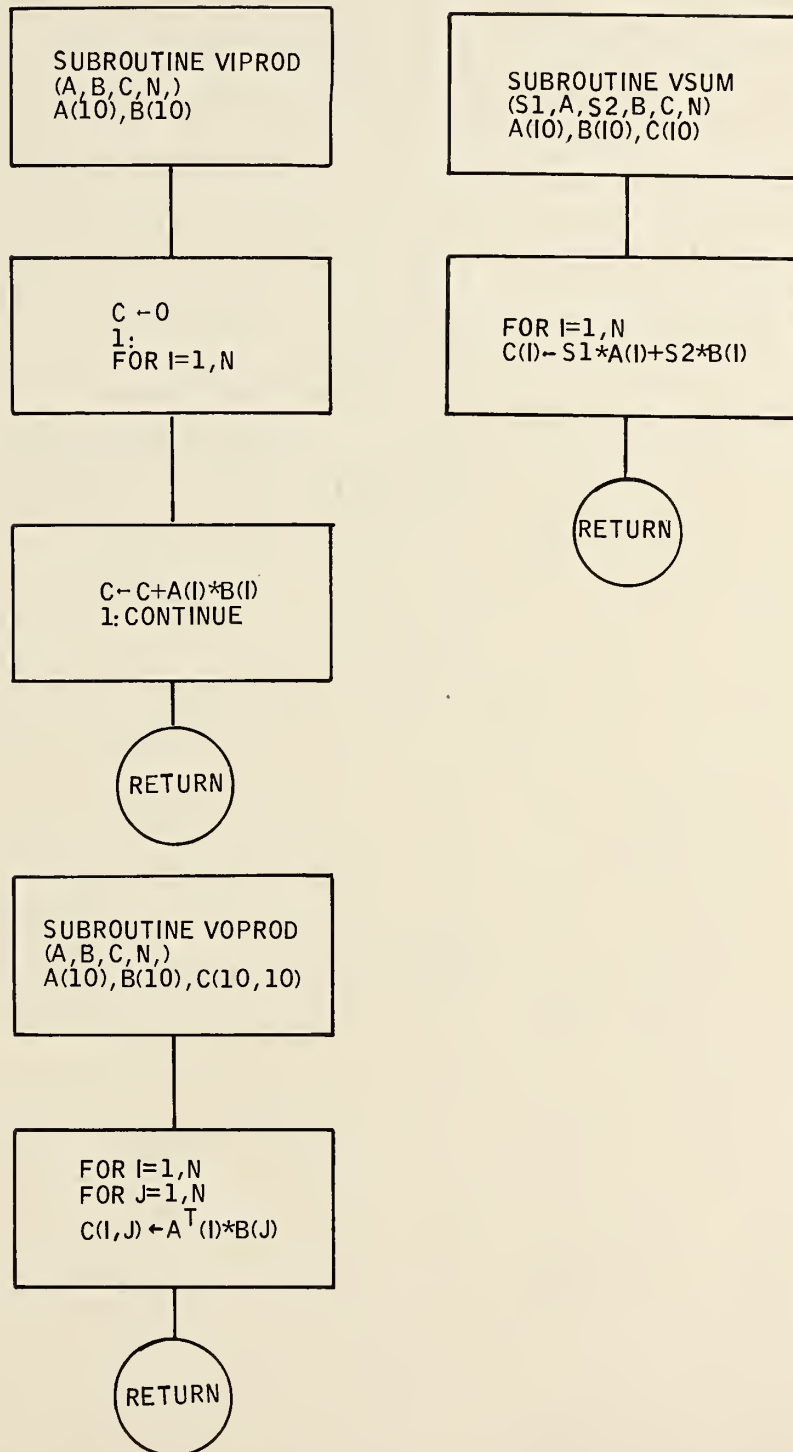


Figure 20. Flow Charts (Continued)

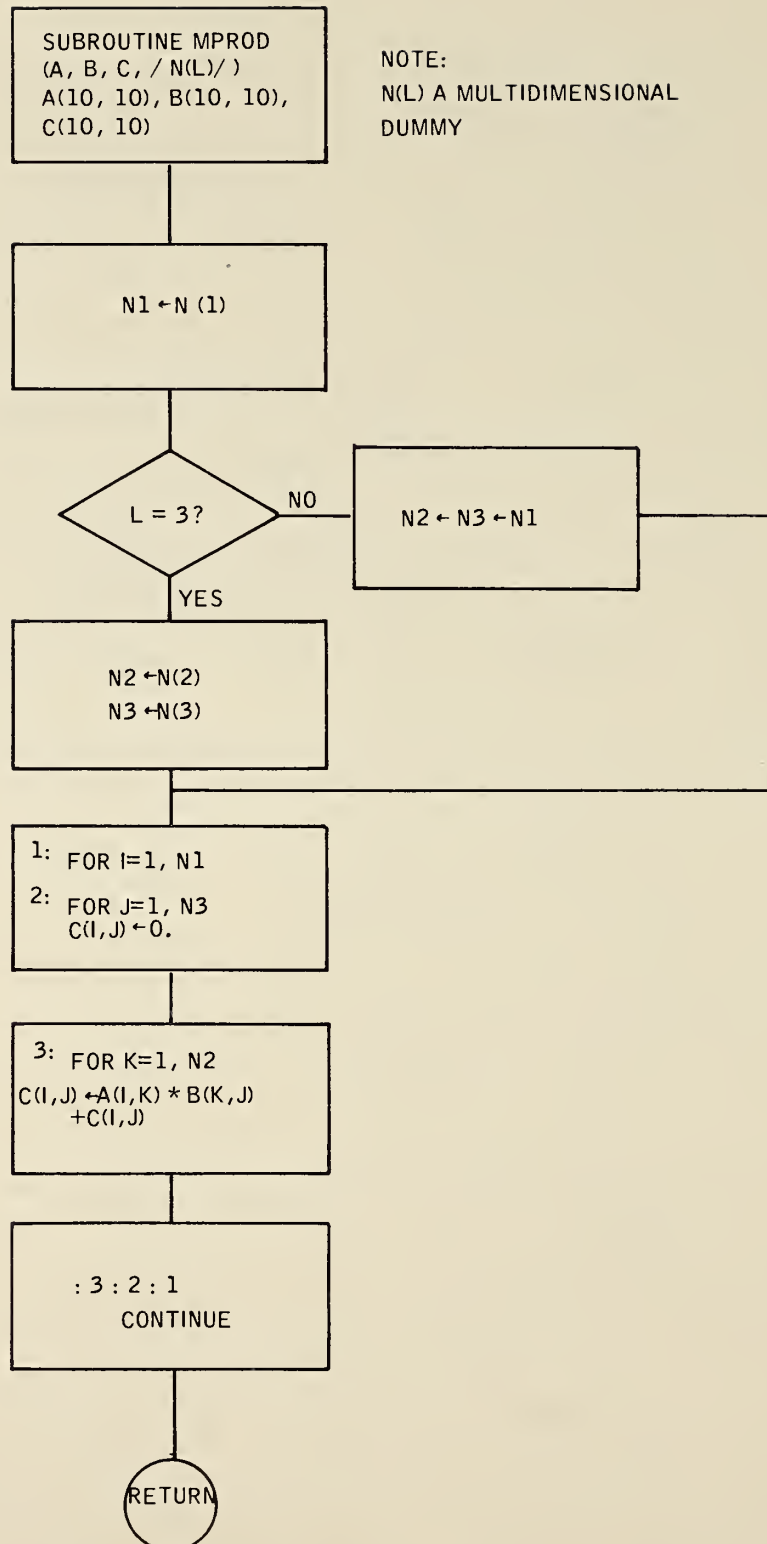


Figure 20. Flow Charts (Continued)



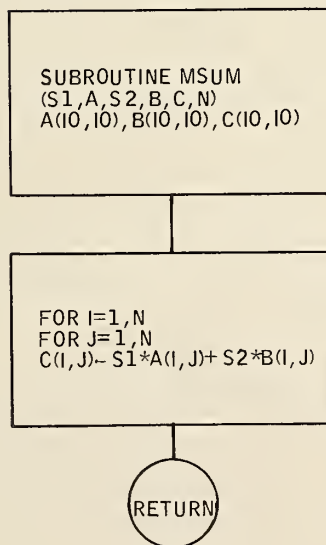
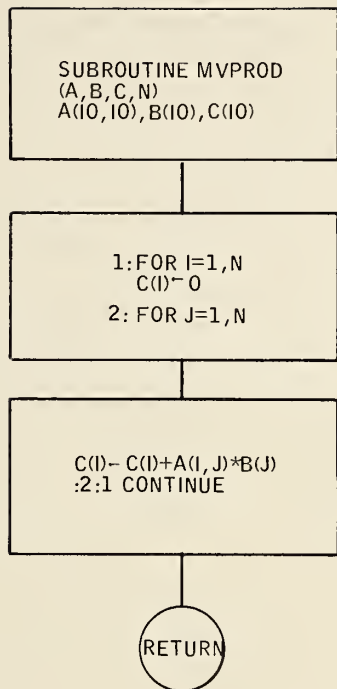


Figure 20. Flow Charts (Continued)

```

SUBROUTINE DMDPROD
(A, B, C, D, N)
A(10), B(10, 10), C(10), D(10, 10)

```

```

FOR I=1, N
FOR J=1, N
D(I, J) = A(I) * B(I, J) * C(J)

```

RETURN

```

SUBROUTINE TRANSP
(A, B, N)
A(10, 10), B(10, 10),

```

```

FOR I=1, N
FOR J=1, N
B(I, J) = A(J, I)

```

RETURN

WORKING SINGLY DIMENSIONED  
ARRAY, DIM (MRA)

TDINVR (IS, IDS, NR, NC, A, MRA, KWA, DET)      OUTPUT VAL OF DETERMINANT

IS-OUTPUT = 1 SOLVED/FOUND  
          = 2 UNABLE TO SOLVE  
          = 3 INPUT ERROR

IDS-OUTPUT = 1 DETERMINANT CALCULATION DID NOT OVERFLOW  
              = 2 DETERMINANT CALCULATION DID OVERFLOW

NR-INPUT      NUMBER OF ROWS IN INPUT MATRIX A  
NC-INPUT      NUMBER OF COLUMNS IN INPUT MATRIX A  
A-INPUT       NOT PRESERVED  
          OUTPUT      INVERSE OF INPUT MATRIX A  
MRA-INPUT     MAX NR OF ROWS IN A

Figure 20. Flow Charts (Continued)

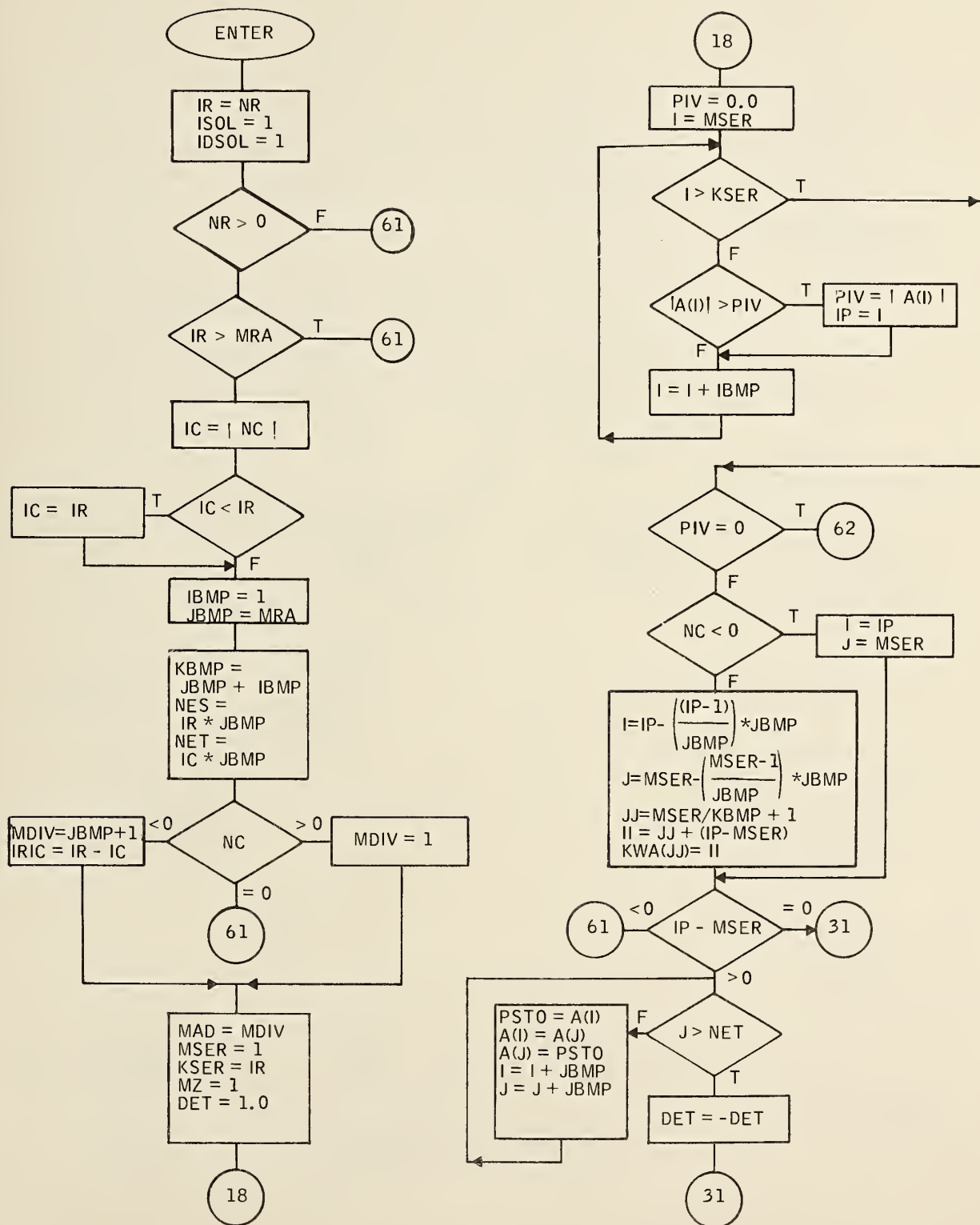


Figure 20. Flow Charts (Continued)

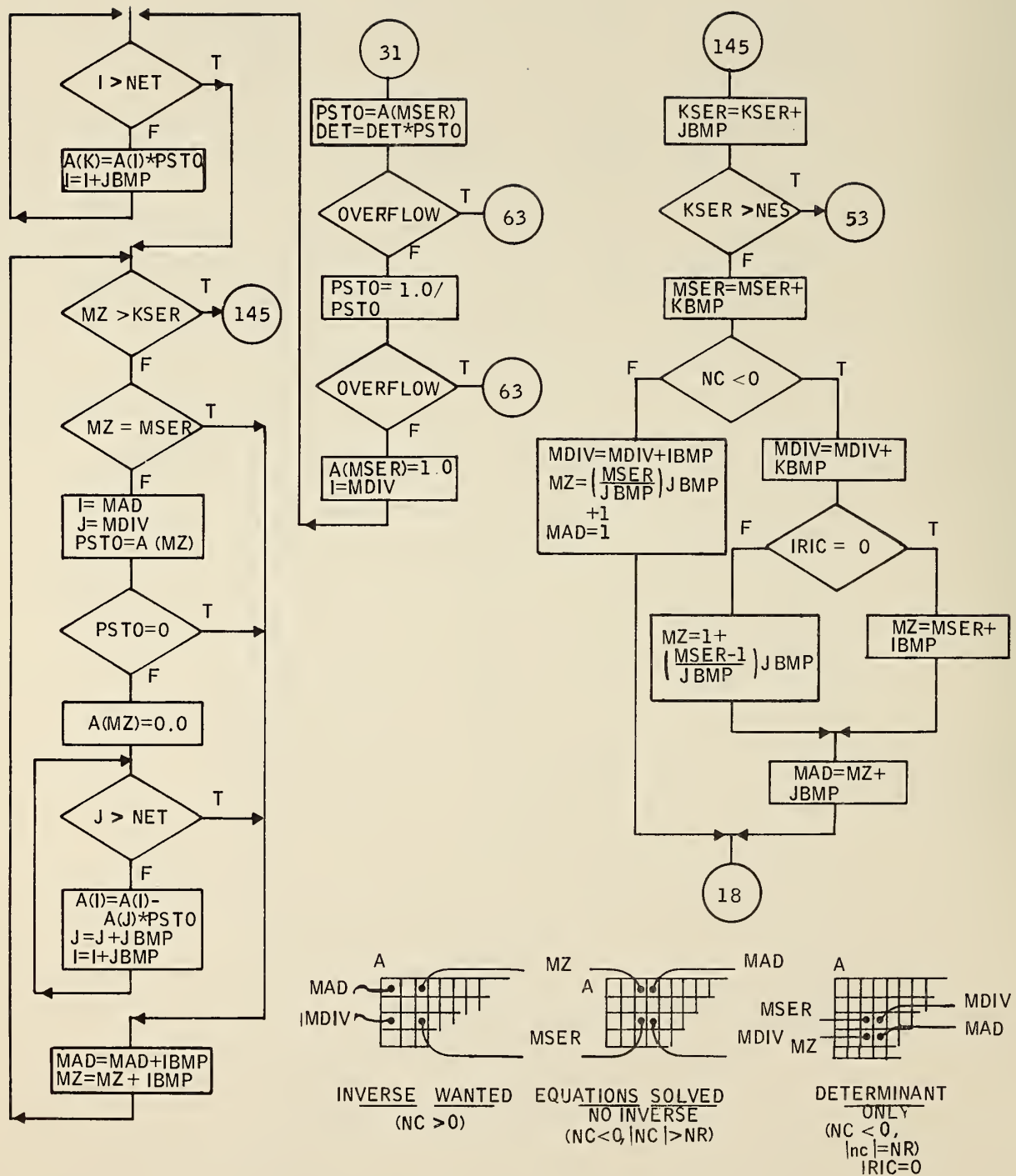


Figure 20. Flow Charts (Continued)

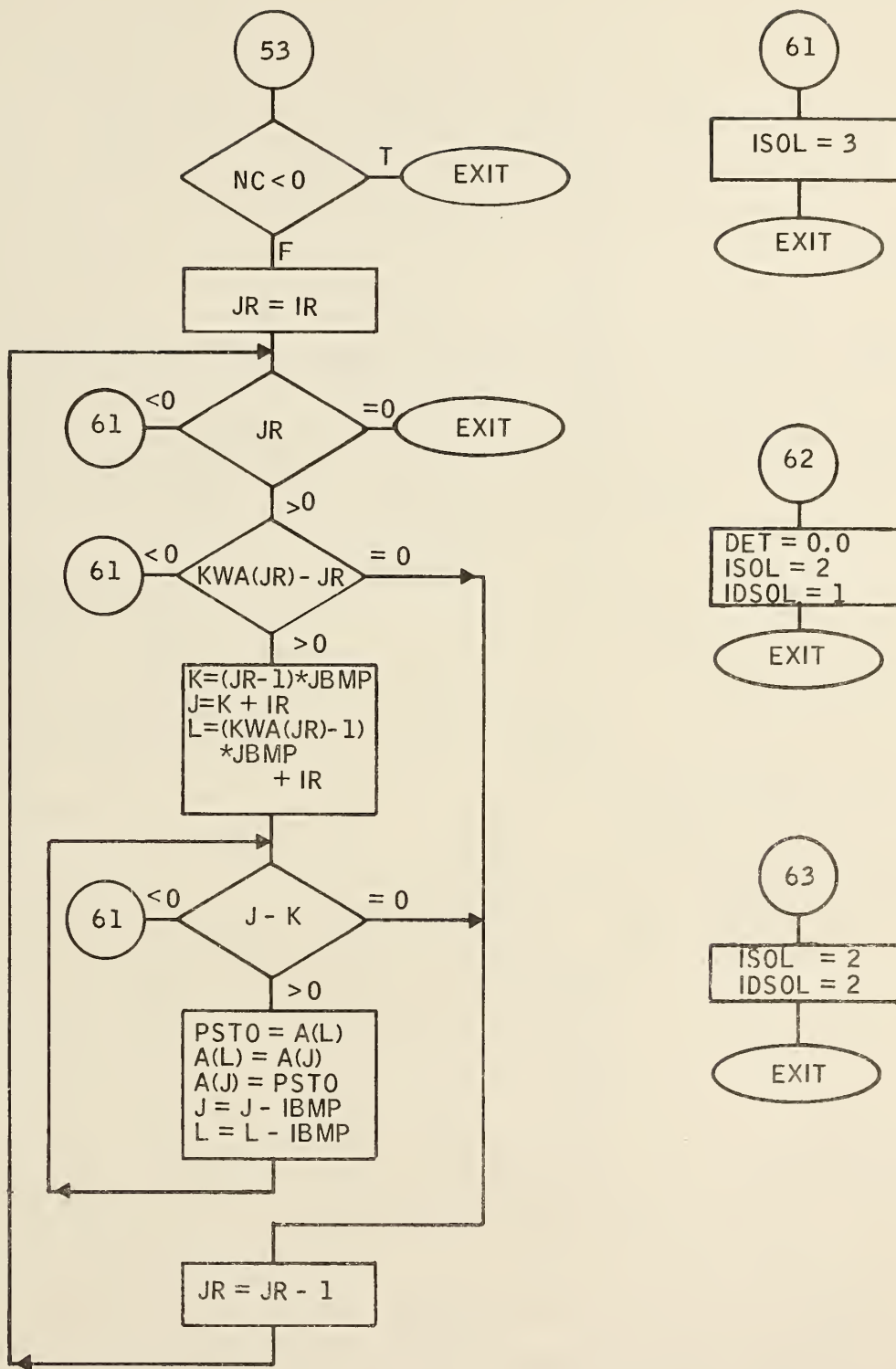


Figure 20. Flow Charts (Concluded)



## CHAPTER 6

### DEVELOPMENT OF A PASSIVE BUS DETECTOR

#### SUMMARY

The classifier algorithm was designed by analyzing the characteristics of recorded vehicle signatures. These signatures were produced by test vehicles driving over experimental installations in Minneapolis (Figure 21) and in Washington, D. C. and by actual street traffic in Washington, D. C. (Figure 22) shows examples of signatures from nine different types of vehicles. A number of experimental parameters that could affect the vehicle signatures were varied during the experimental runs. These include vehicle parameters (speed, centerline displacement, separation, etc.) and installation parameters (asphalt, concrete, reinforcing mesh, lead wire length, snow pack, etc.). Also, signatures were recorded for vehicles that stopped over the sensing element.

A number of signal characteristics were apparent in the initial recordings and these were used to make a preliminary definition of the classifier structure. First, all vehicles produced only positive signal values with peak values related to vehicle mass and inversely related to road clearance. Thus, a signal threshold would detect vehicle presence and would reject most bicycles and motorcycles. Second, vehicles with short wheel bases produce a signal that has only one peak value, while long-wheel-base vehicles show multiple peaks. This suggested that if the multiple peaks were consistent and could be measured, then a multi-stage classifier structure could reduce the real-time data processing rate by counting peaks and rejecting most automobiles. The vehicles that pass this gross test are then classified by using additional, more detailed features. This classifier structure is shown in the sketch that follows:

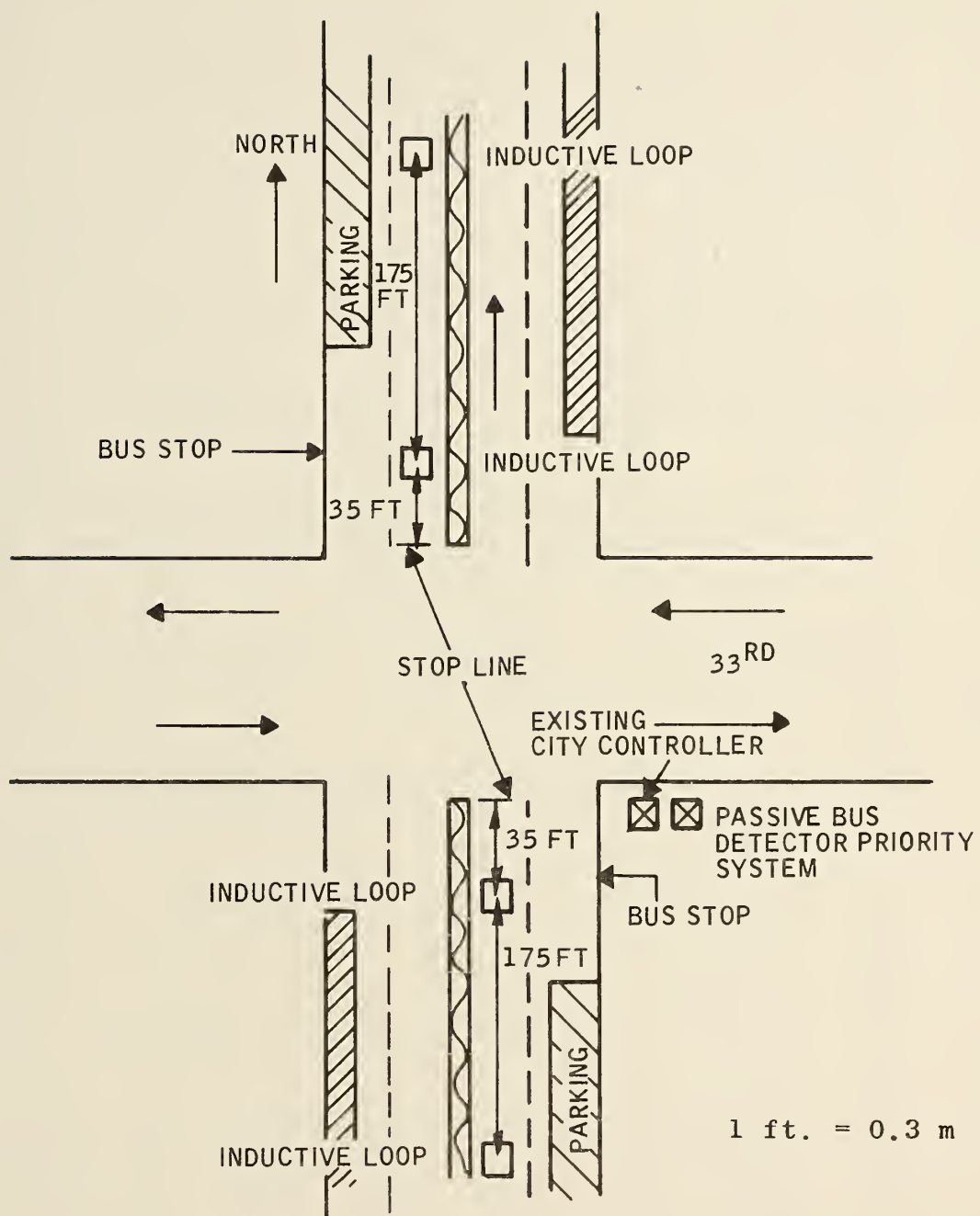


Figure 21. Installation and Demonstration, Minneapolis-33rd and Johnson St. NE

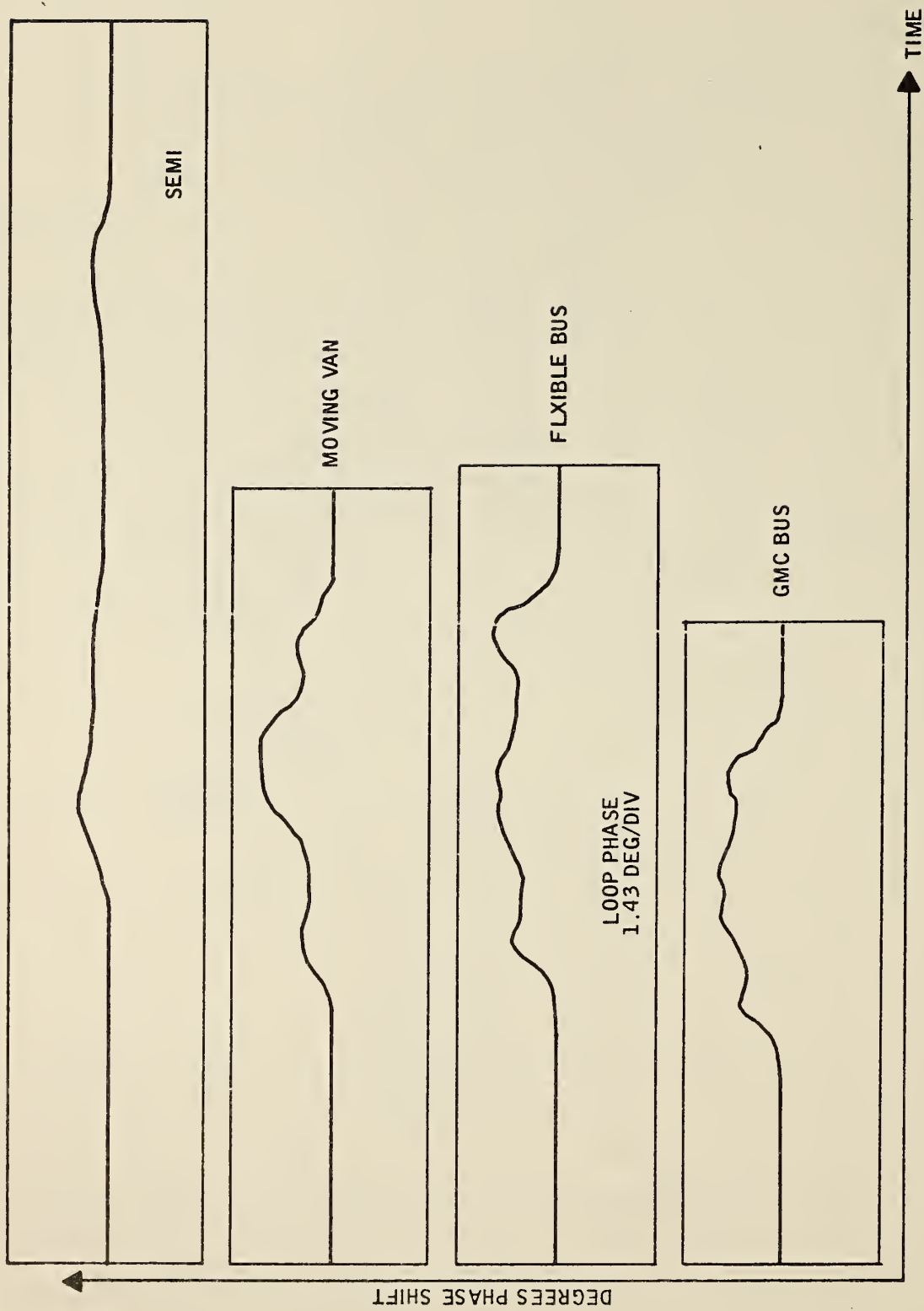


Figure 22a. Vehicle Signature Examples

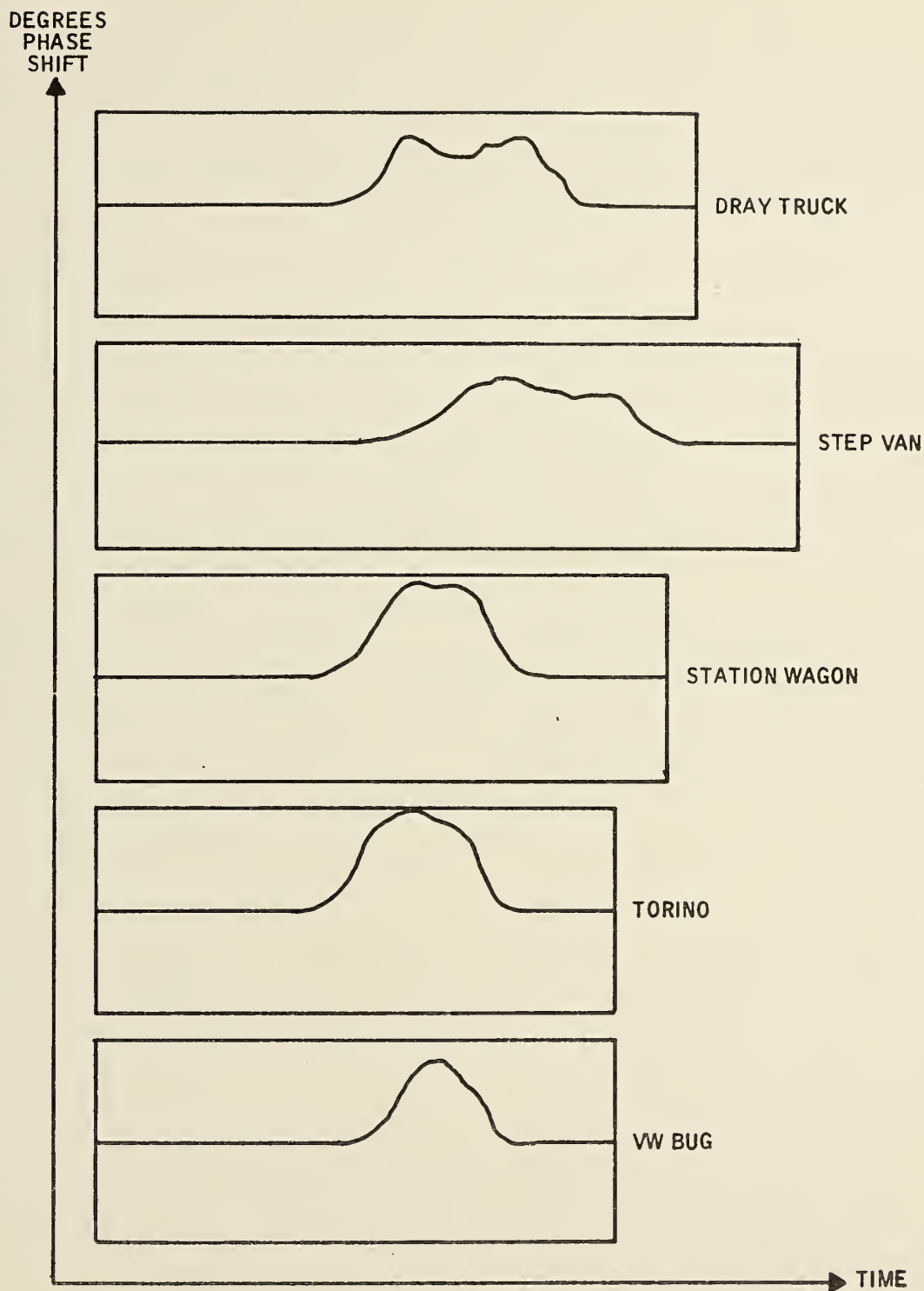
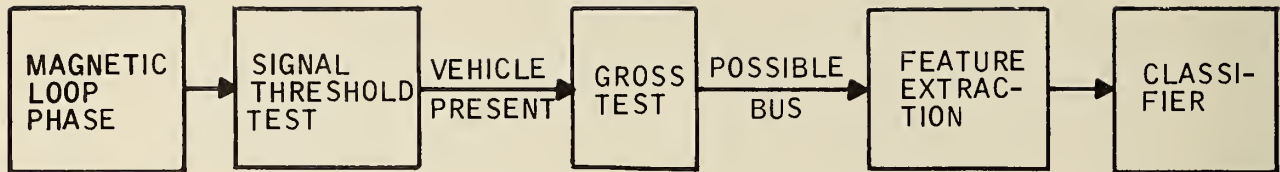


Figure 22b. Vehicle Signature Examples (Concluded)



The design procedure started by implementing the signal threshold test. Having determined the threshold value, the next step was to define an algorithm for detecting signal peaks. A number of algorithms were formulated and tested. The one chosen was able to detect all the major peaks that were obvious to the eye (independent of the shape of the peak) and it detected a minimum number of spurious peaks (small amplitude signal variations and noise effects). The gross test was implemented next. Eyeball study of signal tracings suggested that busses within four feet\* of center line produced from three to six major peak values. This test was implemented and a number of tests were run against recorded data using various lower and upper limits to verify this criterion.

Up to this point preliminary values had been established for the vehicle detection threshold, the peak detection threshold and the upper and lower limits for the gross test. These three processing steps passed all of the bus signals and eliminated more than 90 percent of the non-bus events that would be encountered in normal traffic. What remained to be done was to find appropriate signal measurements (features) that would let us distinguish between busses and the other vehicles that passed the gross test. Moving vans, semis, and non-urban transit busses are examples of the vehicles to be eliminated. The signals from these other vehicles showed roughly the same signal shape as the busses. There was also the problem of signal shapes being distorted by a vehicle changing speed or stopping.

By plotting peak values versus time for a number of different test runs it was seen that these signal characteristics contained enough information for a

\* 1 ft. = 0.3 m



human to discriminate between various vehicles moving with uniform speed. It was decided to attempt a classifier design using the peak values as features. This then led to a decision to use two classifiers, one for uniformly moving vehicles and one for stopping (or non-uniformly moving) vehicles. The time measurements then served to determine which type of motion was being observed and, for uniform speed vehicles, the time measurements would provide a speed estimate and thus allow the effect of speed differences to be removed. The final uncontrolled non-significant variable was the change in signal magnitude caused by the displacement of the vehicle from the loop's center line. It was decided that this effect could be treated as a signal attenuation factor that did not change the signal shape and its effect was removable by normalizing the signal amplitudes.

The amplitudes of the signal peaks were normalized by dividing each peak amplitude by the amplitude of the first peak. Thus, amplitude of the first peak of the normalized signal is always equal to 1. The measurements of peak magnitudes and times (primary features) thus had to be transformed or normalized into values called secondary features that were independent of vehicle speed and lateral displacement. The normalization is described in the following two paragraphs.

To perform the feature analysis and the classifier design, the primary features were punched on cards for input to the XDS9300 digital computer. A number of methods for normalizing the peak magnitudes and times were tested using the primary features from vehicles that passed the gross test. It was found that the ratio of Nth peak magnitude to 1st peak magnitude provided the necessary magnitude normalization. (That is to say, the ratios  $M_1/M_1$ ,  $M_2/M_1$ , ...,  $M_6/M_1$  where  $M_N$  is the Nth peak magnitude.)

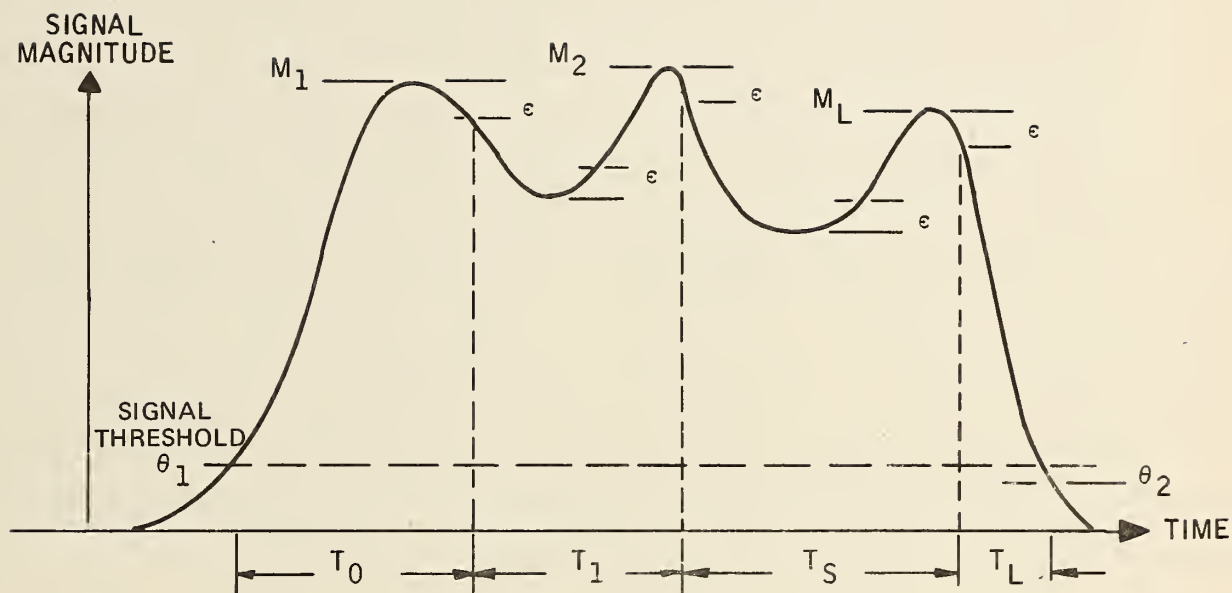
Normalizing the time values was not as easy. A velocity estimate was the ideal normalizing factor, but this required knowing the length of the vehicle and that, in turn, required knowing what kind of vehicle made the signal.

An attempt was made to use the total signal duration, but this caused all vehicles to have the same signal length and produce confusion between long and short vehicles. The normalizing factor finally used was the sum of the time from vehicle detection to first peak plus the time from last peak to end of detection. The primary time features were divided by this factor to produce the (normalized) secondary time features. The normalized time from detection to first peak was found to be near 0.5 for all vehicles moving with uniform velocity, thus the moving versus stopped vehicle test was based on this value. (The test was later modified to separate uniform and non-uniform speeds. The non-uniform speed classifier then was concerned with accelerating and decelerating vehicles as well as stopping vehicles.)

One more decision was made to simplify the classifier structure - that only the first, second and last peaks (magnitude and time) would be used to generate secondary features. Any peaks detected between the second and the last would be ignored.\* Since every signal passing the gross test showed three or more peaks and since all the recorded bus signatures had three major peaks, it was felt that this constraint would not limit the classifier performance. In terms of classifier structure, each primary feature set contained the same number of measurements, regardless of how many peaks were detected. The classifier equations, therefore, did not have to account for the varying number of peaks. The validity of this choice and the normalizations were established by the classifier performance.

---

\*Seven memory locations are used to store the three peak magnitudes and the four time values. Denote these locations by  $(M_1)$ ,  $(M_2)$ ,  $(M_L)$ ,  $(T_0)$ ,  $(T_1)$ ,  $(T_S)$ ,  $(T_L)$ . When the first peak is detected, its values are stored in  $(M_1)$ ,  $(T_0)$ . The second peak is stored in  $(M_2)$ ,  $(T_1)$ . If there are more than two peaks then they are stored in  $(M_L)$ ,  $(T_S)$  with each new peak writing over any information previously stored there. The time at the end of the signal is stored in  $(T_L)$ . Thus, the last detected peak appears in  $(M_L)$ ,  $(T_S)$  at the end of the signal. Figure 23 shows the origins of the seven features from an idealized vehicle signature.



#### PRIMARY FEATURES

- P<sub>1</sub>: T<sub>0</sub> = TIME FROM TURN ON TO FIRST PEAK  
 P<sub>2</sub>: T<sub>L</sub> = TIME FROM LAST PEAK TO TURN OFF  
 P<sub>3</sub>: T<sub>1</sub> = TIME FROM FIRST PEAK TO SECOND PEAK  
 P<sub>4</sub>: T<sub>S</sub> = TIME FROM SECOND PEAK TO LAST PEAK  
 P<sub>5</sub>: M<sub>1</sub> = MAGNITUDE OF FIRST PEAK  
 P<sub>6</sub>: M<sub>2</sub> = MAGNITUDE OF SECOND PEAK  
 P<sub>7</sub>: M<sub>L</sub> = MAGNITUDE OF LAST PEAK

Figure 23. Primary Feature Measurements



The secondary feature set then had two normalized magnitudes and two normalized time intervals. A graphical/statistical analysis was performed to determine whether these secondary features contained enough information to distinguish between busses and other vehicles. For this analysis, the thresholds were set to allow an abnormally large number of non-bus runs to pass the gross test. This provided signatures from the three busses that had been used, plus signatures from six other vehicles (Volkswagen, station wagon, step van, dray truck, semi and moving van). Figure 24 shows the normalized signature averages for these nine vehicle types. Each of these was treated as a separate vehicle class and the sizes and locations of these nine clusters was calculated. The results were that eight clusters were located close to a two-dimensional plane in the four-dimensional space of secondary features (the moving van was the maverick), the three bus clusters and the moving van cluster were close to each other but far from the other five clusters. Figure 25 is a polar plot of the nine different vehicle class means.

#### DESIGN OF THE MOVING VEHICLE CLASSIFIER

With this evidence in favor of the four secondary features being sufficient, the next step was to determine equations for a classifier. The method used was to arbitrarily assign a number to each cluster of data samples (for example: 1 denotes bus, 2 denotes semi, 3 denotes truck, etc.). These are called "class values". Then, considering the class values as a function defined at the sampled points in the four-dimensional space of secondary features, a linear function (called a "Discriminant") is found that gives the best least-squares fit to the class values. (Ideally this function will be near 1 for all the points in cluster number 1, near 2 in cluster 2, etc.) For the general multiclass problem (more than two classes) this method requires that the class values be permuted to determine the minimum least-square error, but our problem involved only two classes (bus and not bus) so this step was not necessary. Having determined the discriminant, the classification decision was made by comparing the function value with a fixed threshold

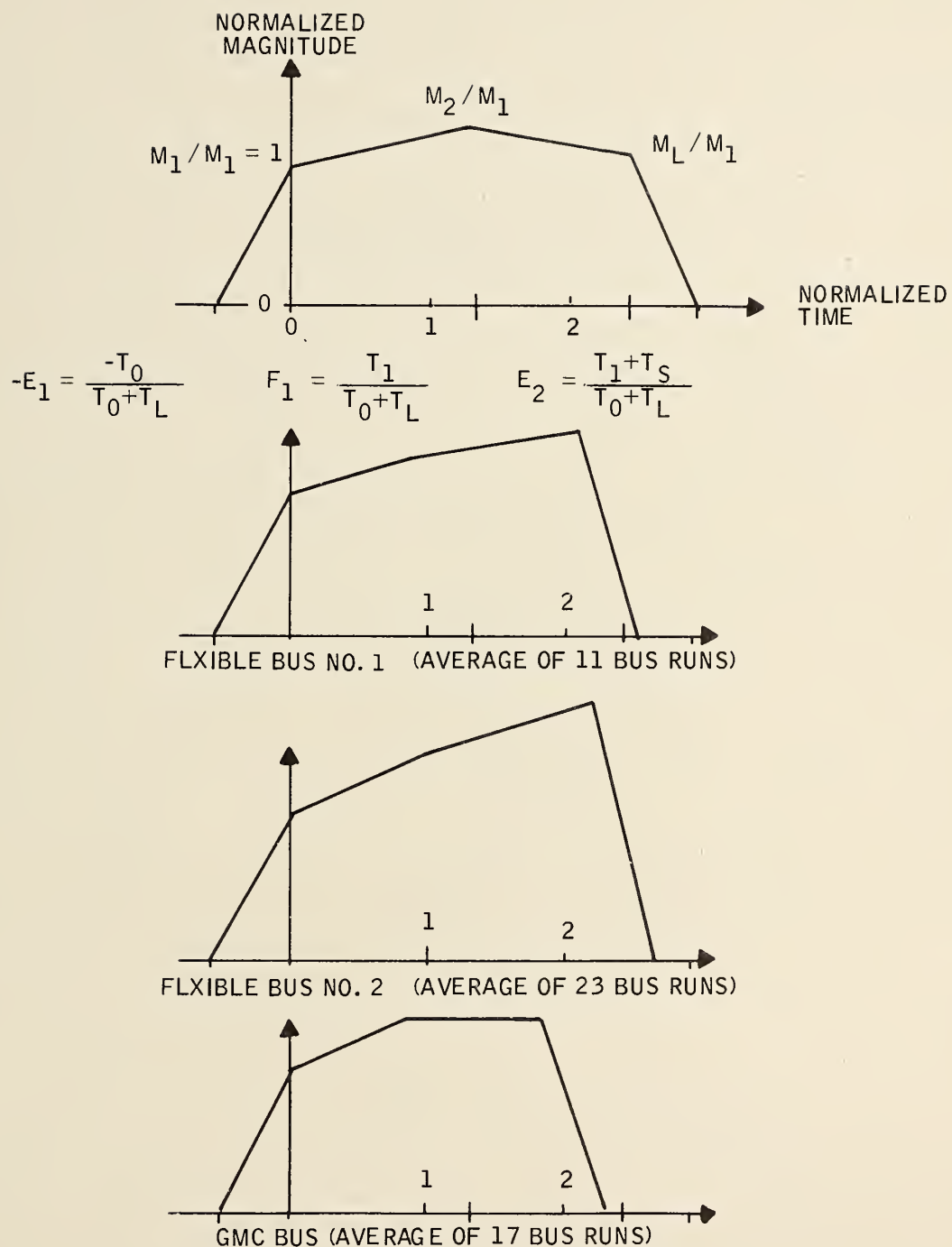


Figure 24a. Normalized Signature Average



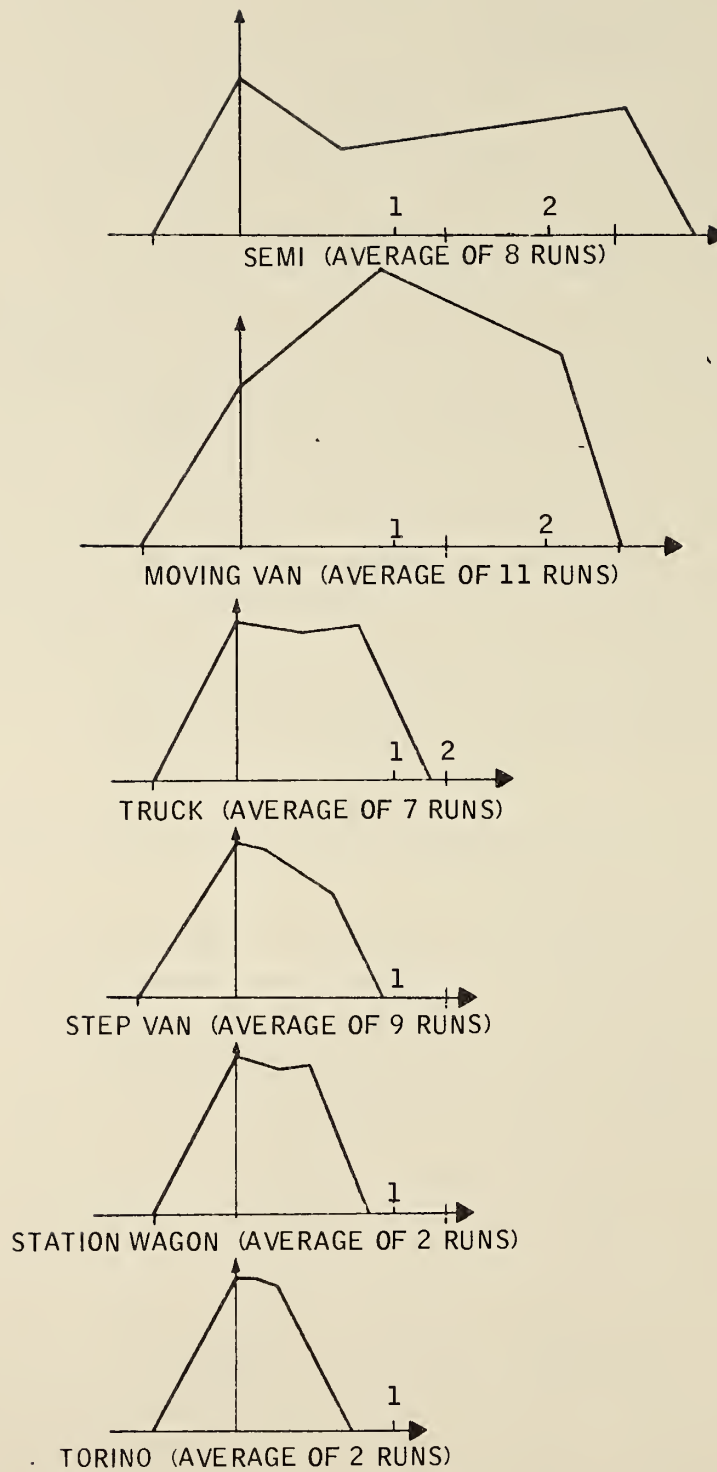


Figure 24b. Normalized Signature Average (Concluded)

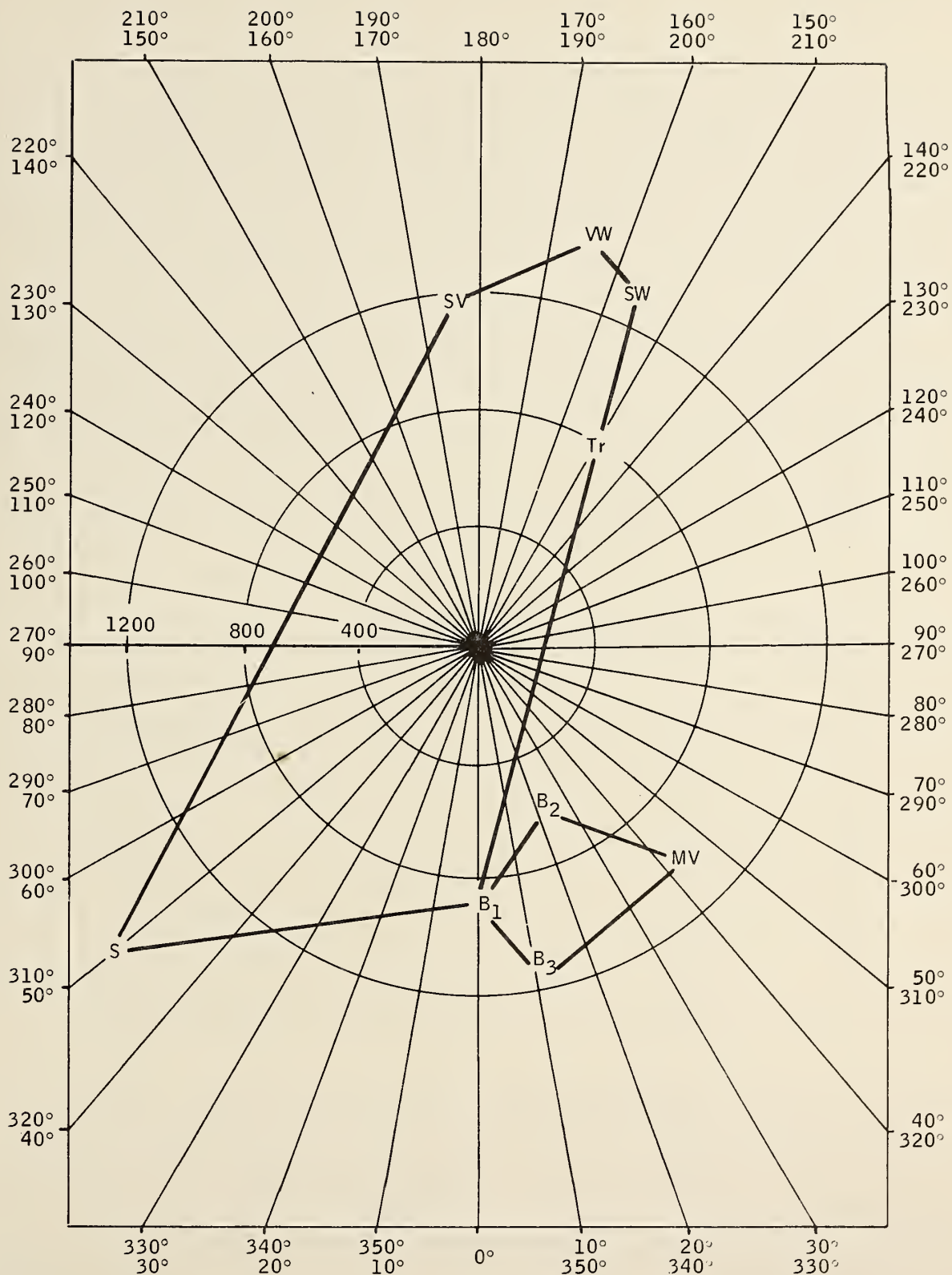


Figure 25. Polar Plot of Subclass Means

value; small values give one decision, and large values the other. For the bus classifier it was not possible to find a single discriminant that would correctly classify all of the experimental data. In the two best cases, one of them classified some of the moving van examples as bus and the other called all of the trucks busses. However, the combination decision rule, that decides "bus" when both discriminants indicate "bus", correctly classified all of the experimental data. Figure 26 shows the two-dimensional distribution of the test data. The region to the left of 2.89 and below 1.57 is the bus region. Neither threshold by itself can separate the bus samples from the others.

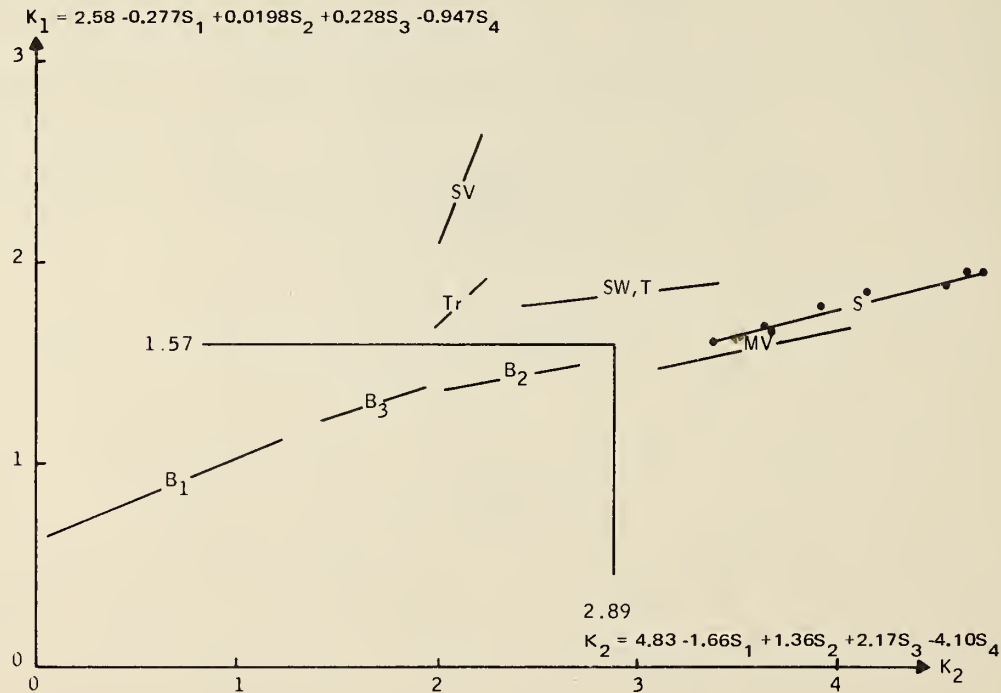


Figure 26. Joint Plot of Moving Vehicle Discriminant Values

The above step in the bus detector data analysis was performed using the CTC classifier program. Feature vector inputs comprised of "eleven" and "twelve" cards and the primary-to-secondary feature transformation code in the main program (controlled by a "minus six" card) were used. Several runs were made initially, using various combinations of selected features

and type-to-class assignments, to determine the basic data distributions and possible discriminants. From these, discriminants and thresholds were selected to be used in the logical mode classifier. The logical mode classifier was executed using five logical variables and nine logical variables, and Karnaugh maps were made from the output of each execution.

Analog data were recorded from the buried loop detector for various vehicles, including buses, vans, trucks, and large and small automobiles. A type was assigned to each vehicle and was recorded at the time of the vehicle's passing the loop. The resulting analog data were transformed into digital data (primary features) and placed on "eleven" and "twelve" cards to be used by the program. Vehicle types were defined as follows:

<u>Vehicle</u>	<u>Type (Moving)</u>	<u>Type (Stopped over the loop)</u>
Flxible Bus 1	1	11
GMC Bus	2	12
Flxible Bus 2	3	13
Semi	4	14
Truck	5	15
Step Van	6	16
Moving Van	7	17
Station Wagon	8	18
Torino	9	19
Volkswagen	10	20

Each feature vector card pair contained seven primary features, along with data describing the road surface, vehicle speed, displacement from loop centerline, vehicle type, trial number, and run number.

The program includes a block of code to transform primary bus detector features into secondary features which are normalized features or combinations of features. These secondary features are "best" for the problem of



vehicle classification in this application. Two transformations are available which differ in the normalizing factors used; control of which transformation is applied is through the use of the "minus six" control card, described elsewhere. Table 40 shows the seven primary features and the two transformations available. In this table,  $T_0$  is the time from vehicle entrance into the loop detection region until the first amplitude peak in detector output,  $T_L$  is the time from the last amplitude peak until vehicle exit from loop detection region,  $T_1$  is the time between the first and second amplitude peaks, and  $T_S$  is the time between the second and last amplitude peaks. Also,  $M_1$ ,  $M_2$ , and  $M_L$  are the amplitude values of the first, second, and last amplitude peaks, respectively.

Secondary feature  $S_1$  is the normalized time between the first and second amplitude peaks,  $S_2$  is the normalized time between the second and last peaks,  $S_3$  is the normalized amplitude of the second peak, and  $S_4$  is the normalized amplitude of the last peak. These four secondary features can be used to classify uniformly moving vehicles. Secondary features  $S_6$  and  $S_7$  are the normalized time from vehicle entry into the loop detection region until the first amplitude peak and the normalized time between the first and last peaks, respectively. These two features can be used to determine whether or not a vehicle is moving. The factor "1000" appearing in these normalized features is necessary because the program will use only the integer part of any feature value.

In order to determine linear discriminants and thresholds which could be used with the logical mode classifier, several runs were made using the method one classifier. One of these runs used secondary features 1, 2, 3, and 4, and the MODE 2 = 1 transformation. For this run, types 1, 2, and 3 (moving buses) were lumped together in class 1; types 4, 5, 6, 7, 8, 9, and 10 (other moving vehicles) were called class 2; and all non-moving vehicles (types 11 through 20) were assigned to no class, which shows as class 0 in the execution. Class 1 and class 2 were assigned equal a-priori probabilities of 0.50, and the method 1 classifier was executed on the feature vector set.



TABLE 40. PRIMARY AND SECONDARY FEATURES  
AVAILABLE IN CTC

Primary Features						
$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$	$P_7$
$T_0$	$T_L$	$T_1$	$T_S$	$M_1$	$M_2$	$M_L$
Secondary Features						
For Mode 2 = 1				For Mode 2 = 2		
$\frac{T_1 \cdot 1000}{T_0 + T_L}$			$S_1$	$\frac{T_1 \cdot 1000}{T_L}$		
$\frac{T_S \cdot 1000}{T_0 + T_L}$			$S_2$	$\frac{T_S \cdot 1000}{T_L}$		
$\frac{M_2 \cdot 1000}{M_1}$			$S_3$	$\frac{M_2 \cdot 1000}{M_1}$		
$\frac{M_L \cdot 1000}{M_1}$			$S_4$	$\frac{M_L \cdot 1000}{M_1}$		
$T_1 + T_S$			$S_5$	$T_1 + T_S$		
$\frac{T_0 \cdot 1000}{T_0 + T_L}$			$S_6$	$\frac{T_0 \cdot 1000}{T_L}$		
$\frac{(T_1 + T_S) \cdot 1000}{T_0 + T_L}$			$S_7$	$\frac{(T_1 + T_S) \cdot 1000}{T_L}$		

The result of this run was a linear discriminant for moving busses versus other moving vehicles (see Table 41).

Although the other 0 vectors were not classified, and occurred throughout the range of scores formed by the discriminant, all moving buses had scores less than 1.471, while all other moving vehicles had scores greater than this value. Thus a linear discriminant which separates moving vehicles into busses or nonbusses perfectly (with this training set of feature vectors) has been found, and by selecting a threshold near 1.471, a logical variable may be formed which is false for any moving bus and true for any moving nonbus. The threshold which was actually chosen was not 1.471 but 1.57; this threshold causes four moving nonbusses of the moving van variety to be called class 1 (moving busses). This seemingly nonoptical threshold was chosen because of the closeness of moving vans to busses in the decision space and to prevent a moving bus from being rejected in a real-world situation, at the expense of a few misclassified moving vans. Another discriminant may be used to make the choice between bus or moving van using a decision space in which these two types are farther apart.

Such a decision space was found in a different execution of the method 1 classifier; this run was similar to that previously described, with the mode 2 = 1 secondary feature transformation and use of secondary features 1, 2, 3, and 4. However, instead of only two classes of interest, five were used as follows:

<u>Class</u>	<u>Contained Types</u>	<u>Names</u>
1	1, 2, 3	Moving busses
2	4	Moving semis
3	5, 6, 8, 9	Moving trucks, moving step vans, moving station wagons, moving Torino cars
4	7	Moving moving van
5	10	Moving Volkswagen

TABLE 41. TRADEOFF LISTING FOR BUS CLASSIFIER DESIGN

DATE 1223 ID 30 RUN9  
TRADEOFF FOR  
PAIR 1 2

COEFF	2.58009	-.00023	.00002	.00023	-.00025			
	NB	ID TY		CL	SCORE	0	1	2
1	64	298 16 51	2 121310	0	-2.4975	1		
2	34	225 12 101	0 121310	1	-.0737	2		
3	116	225 12 101	0 121904	1	.0227	3		
4	96	359 13 51	0 121308	1	.1796	4		
5	94	357 3 51	-3 121308	1	.6727		1	
6	82	344 3 151	3 121308	1	.7249		2	
7	90	352 3 51	4 121308	1	.8218		3	
8	75	337 3 51	6 121308	1	.8495		4	
9	79	341 3 101	3 121308	1	.8580		5	
10	91	353 3 51	-2 121308	1	.8436		6	
11	84	346 3 51	0 121308	1	.8724		7	
12	74	336 3 51	3 121308	1	.9054		8	
13	89	351 3 51	2 121308	1	.9296		9	
14	80	342 3 101	-3 121308	1	.9336		10	
15	93	356 3 51	3 121308	1	.9395		11	
16	76	338 3 51	-3 121308	1	.9485		12	
17	95	358 13 51	0 121308	0	.9506	5		
18	88	350 3 51	2 121308	1	.9230		13	
19	83	345 3 151	-3 121308	1	.9345		14	
20	92	355 3 51	0 121308	1	1.0158		15	
21	72	334 3 51	0 121308	1	1.0257		16	
22	81	343 3 151	0 121308	1	1.0301		17	
23	73	335 3 51	0 121308	1	1.0537		18	
24	78	340 3 101	0 121308	1	1.0555		19	
25	77	339 3 51	-6 121308	1	1.0721		20	
26	85	347 3 51	0 121308	1	1.0745		21	
27	38	240 2 101	6 121310	1	1.0782		22	
28	31	221 2 101	6 121310	1	1.0830		23	
29	86	348 3 51	0 121308	1	1.0868		24	
30	87	349 3 51	0 121308	1	1.1144		25	
31	67	305 16 51	-4 121904	0	1.1233	6		
32	99	99 1 101	0 121903	1	1.2069		26	
33	8	105 1 101	-6 121310	1	1.2120		27	
34	2	99 1 101	0 121310	1	1.2254		28	
35	5	101 1 101	0 121310	1	1.2340		29	
36	33	224 12 101	0 121310	0	1.2582	7		
37	6	102 1 101	3 121310	1	1.2629		30	
38	3	995 1 101	0 121310	1	1.2875		31	
39	100	995 1 101	0 121903	1	1.2914		32	
40	101	100 1 101	0 121903	1	1.3021		33	
41	7	104 1 101	-3 121310	1	1.3033		34	
42	4	100 1 101	0 121310	1	1.3039		35	
43	32	222 2 101	-3 121310	1	1.3219		36	
44	102	101 1 101	0 121903	1	1.3273		37	
45	40	243 12 101	0 121310	0	1.3289	8		
46	39	241 2 101	-3 121310	1	1.3588		38	
47	27	217 2 101	0 121310	1	1.3915		39	
48	37	239 2 101	3 121310	1	1.4024		40	
49	30	220 2 101	3 121310	1	1.4049		41	
50	112	217 2 101	0 121904	1	1.4061		42	
51	115	220 2 101	3 121904	1	1.4066		43	
52	26	216 2 101	0 121310	1	1.4237		44	
53	28	218 2 101	0 121310	1	1.4422		45	

TABLE 41. TRADEOFF LISTING FOR BUS CLASSIFIER  
DESIGN (CONCLUDED)

54	113	218	2	101	0	121904	1	1.4499	46
55	35	237	2	101	0	121310	1	1.4514	47
56	111	216	2	101	0	121904	1	1.4537	48
57	29	219	2	101	0	121310	1	1.4640	49
58	36	238	2	101	0	121310	1	1.4664	50
59	114	219	2	101	0	121904	1	1.4703	51
60	97	246	7	101	0	121310	2	1.4730	1
61	44	250	7	101	6	121310	2	1.4989	2
62	66	304	16	51	-4	121904	0	1.4997	9
63	42	246	7	101	0	121310	2	1.5275	3
64	43	247	7	101	0	121310	2	1.5538	4
65	117	246	7	101	0	121904	2	1.5709	5
66	41	245	7	101	0	121904	2	1.5742	6
67	118	250	7	101	6	121904	2	1.5998	7
68	16	167	4	101	3	121310	2	1.6014	8
69	46	254	7	102	0	121310	2	1.6113	9
70	98	247	7	101	0	121310	2	1.6343	10
71	19	171	4	101	-6	121310	2	1.6570	11
72	18	170	4	101	-4	121310	2	1.6626	12
73	105	205	5	101	0	121903	2	1.6792	13
74	20	205	5	101	0	121310	2	1.6819	14
75	45	251	7	101	6	121904	2	1.6868	15
76	23	208	5	101	0	121310	2	1.6959	16
77	65	300	16	51	4	121904	0	1.6998	10
78	22	207	5	101	0	121310	2	1.7015	17
79	24	212	5	101	-6	121310	2	1.7122	18
80	63	296	16	51	0	121310	0	1.7249	11
81	57	278	16	52	-4	121310	0	1.7756	12
82	14	165	4	101	0	121903	2	1.7797	19
83	11	157	8	101	0	121903	2	1.7859	20
84	12	158	8	101	0	121903	2	1.8069	21
85	9	148	9	101	0	121903	2	1.8227	22
86	15	166	4	101	0	121903	2	1.8410	23
87	13	1645	4	101	0	121903	2	1.8653	24
88	21	206	5	101	0	121903	2	1.8854	25
89	25	213	5	101	0	121904	2	1.8901	26
90	10	149	9	101	0	121904	2	1.8982	27
91	53	274	16	52	4	121310	0	1.9035	13
92	1	0	10	0	0	121903	2	1.9255	28
93	17	169	4	101	-3	121310	2	1.9272	29
94	103	169	4	101	-3	121903	2	1.9275	30
95	62	295	16	51	0	121310	0	1.9392	14
96	106	254	7	102	0	121903	2	2.0654	31
97	71	317	6	151	3	121904	2	2.0813	32
98	69	312	6	51	0	121310	2	2.0873	33
99	70	316	6	151	0	121310	2	2.0892	34
100	68	307	6	21	3	121904	2	2.0950	35
101	61	289	6	101	3	121310	2	2.0957	36
102	48	264	6	102	3	121310	2	2.1204	37
103	56	277	16	52	-2	121310	0	2.1434	15
104	107	262	6	102	0	121903	2	2.1438	38
105	121	274	16	52	4	121904	0	2.1501	16
106	55	276	16	52	-2	121310	0	2.1713	17
107	119	262	6	102	0	121904	2	2.1719	39
108	47	262	6	102	0	121310	2	2.1751	40
109	110	273	16	52	2	121903	0	2.1796	18
110	52	273	16	52	2	121310	0	2.1862	19
111	49	270	16	52	0	121310	0	2.2107	20
112	108	270	16	52	0	121903	0	2.2111	21
113	54	275	16	52	4	121904	0	2.2141	22
114	51	272	16	52	2	121310	0	2.2202	23
115	60	283	16	52	-2	121310	0	2.3217	24
116	58	280	16	52	0	121310	0	2.3274	25
117	59	281	16	52	0	121310	0	2.3292	26
118	109	271	16	52	0	121903	0	2.6039	27
119	120	271	16	52	0	121904	0	2.6084	28
120	50	271	16	52	0	121310	0	2.6145	29
121	104	0	10	0	0	121903	2	2.7030	41



Types 11 through 20 were not assigned to any class and are thus in class 0. Each of the five classes was assigned an a-priori probability of 0.20.

The output from this run included linear discriminants for each pairwise combination of the five classes; the discriminant of concern is between class 1 and class 4 (busses versus moving vans). In the tradeoff listing of scores for this discriminant (see Table 42), all class 1 vectors have scores of 2.6667 or less, while all class 4 vectors had scores of 3.1082 or more. A threshold between these values was selected, 2.890. Since there is a wider separation between busses and moving vans in this decision space, the safety margin (or margin of allowed errors) which was necessary in the previous run was not required. The mixing of classes 2, 3, and 5 throughout the tradeoff listing is of little concern, since the goal is to select between classes 1 and 4. Application of the first discriminant will select all moving busses and some moving vans. Application of the second discriminant to this group of busses and moving vans will select the busses and reject the moving vans.

## DESIGN OF THE STOPPED VEHICLE CLASSIFIER

The preceding discussion refers to the moving vehicle classifier. For the stopped vehicles, only the normalized signal magnitude features were used. For a stopped vehicle, one of the time intervals becomes arbitrarily large and thus does not characterize the vehicle class. The method for finding discriminants was the same as described above, but the decision rule was slightly different. In this two-dimensional space the clusters all lay near one line with the nonbusses at the outside and the busses in the middle. Thus an upper and a lower threshold on the discriminant values was needed. Again, two discriminants were combined to correctly classify all of the experimental data. Figure 27 shows how the stopped vehicle discriminant values and the thresholds are arranged.



TABLE 42. TRADEOFF LISTING FOR BUS CLASSIFIER  
DESIGN

DATE 1223 ID 80 RUN7  
TRADEOFF FOR  
PAIR 1 4

COEFF		ID TY			CL		SCORE	0	1	2	3	4	5
4.83403		298	16	51	2	121310	0	-15.4957	1				
1 64		225	12	101	0	121310	0	-6.4864	2				
2 34		225	12	101	0	121904	0	-5.7883	3				
3 116		359	13	51	0	121308	0	-4.3853	4				
4 96		305	16	51	-4	121904	0	-7.7847	5				
5 67		357	3	51	-3	121308	1	-3.3248		1			
6 94		344	3	151	3	121308	1	0.658		2			
7 82		353	3	51	-2	121308	1	0.895		3			
8 91		352	3	51	4	121308	1	0.912		4			
9 90		337	3	51	6	121308	1	1.244		5			
10 75		346	3	51	0	121308	1	3.674		6			
11 84		356	3	51	3	121308	1	3.825		7			
12 93		341	3	101	3	121308	1	4.087		8			
13 79		336	3	51	3	121308	1	5.397		9			
14 74		342	3	101	-3	121308	1	5.677		10			
15 80		338	3	51	-3	121308	1	6.078		11			
16 76		351	3	51	2	121308	1	6.218		12			
17 89		240	2	101	6	121310	1	6.457		13			
18 38		221	2	101	6	121310	1	6.545		14			
19 31		345	3	151	-3	121308	1	7.075		15			
20 83		339	3	51	-6	121308	1	7.943		16			
21 77		350	3	51	2	121308	1	8.310		17			
22 88		355	3	51	0	121308	1	9.209		18			
23 92		343	3	151	0	121308	1	9.360		19			
24 81		334	3	51	0	121308	1	9.540		20			
25 72		340	3	101	0	121308	1	9.840		21			
26 78		335	3	51	0	121308	1	10.360		22			
27 73		105	1	101	-6	121310	1	10.962		23			
28 8		347	3	51	0	121308	1	11.141		24			
29 85		348	3	51	0	121308	1	11.923		25			
30 86		349	3	51	0	121308	1	12.102		26			
31 87		101	1	101	0	121310	1	13.927		27			
32 5		99	1	101	0	121903	1	15.511		28			
33 99		99	1	101	0	121310	1	15.578		29			
34 2		995	1	101	0	121903	1	18.137		30			
35 100		995	1	101	0	121310	1	18.175		31			
36 3		104	1	101	-3	121310	1	18.883		32			
37 7		100	1	101	0	121310	1	18.842		33			
38 4		100	1	101	0	121903	1	18.879		34			
39 101		102	1	101	3	121310	1	19.201		35			
40 6		205	5	101	0	121903	3	20.070			1		
41 105		205	5	101	0	121310	3	20.172			2		
42 20		208	5	101	0	121310	3	20.443			3		
43 23		222	2	101	-3	121310	1	20.530	36				
44 32		207	5	101	0	121310	3	20.899			4		
45 22		217	2	101	0	121904	1	20.959	37				
46 112		241	2	101	-3	121310	1	21.100	38				
47 39		217	2	101	0	121310	1	21.740	39				
48 27		239	2	101	3	121310	1	21.856	40				
49 37		212	5	101	-6	121310	3	22.129			5		
50 24		220	2	101	3	121310	1	22.549	41				
51 30		220	2	101	3	121904	1	22.817	42				
52 115		216	2	101	0	121310	1	22.957	43				
53 26													

TABLE 42. TRADEOFF LISTING FOR BUS CLASSIFIER  
DESIGN (CONCLUDED)

54	102	101	1	101	0	121903	1	2.3682	44		
55	113	218	2	101	0	121904	1	2.3764	45		
56	111	216	2	101	0	121904	1	2.3949	46		
57	11	157	8	101	0	121903	3	2.4235		6	
58	28	218	2	101	0	121310	1	2.4760	47		
59	35	237	2	101	0	121310	1	2.5837	48		
60	29	219	2	101	0	121310	1	2.6377	49		
61	12	158	8	101	0	121903	3	2.6482		7	
62	114	219	2	101	0	121904	1	2.6608	50		
63	36	238	2	101	0	121310	1	2.6667	51		
64	9	148	9	101	0	121903	3	2.6705		8	
65	44	250	7	101	6	121310	4	3.1782			4
66	106	254	7	102	0	121903	4	3.2765			2
67	43	247	7	101	0	121310	4	3.3182			3
68	10	149	9	101	0	121904	3	3.3681		9	
69	16	167	4	101	3	121310	2	3.3694		1	
70	117	246	7	101	0	121904	4	3.3716			4
71	95	358	13	51	0	121308	0	3.4040	6		
72	41	245	7	101	0	121904	4	3.4407			5
73	97	246	7	101	0	121310	4	3.5644			6
74	53	274	16	52	4	121310	0	3.6009	7		
75	65	300	16	51	4	121904	0	3.6260	8		
76	18	170	4	101	-4	121310	2	3.6307		2	
77	62	295	16	51	0	121310	0	3.6522	9		
78	19	171	4	101	-6	121310	2	3.6718		3	
79	118	250	7	101	6	121904	4	3.7327			7
80	45	251	7	101	6	121904	4	3.8122			2
81	98	247	7	101	0	121310	4	3.8230			3
82	14	165	4	101	0	121903	2	3.9243		4	
83	21	206	5	101	0	121903	3	4.0186		10	
84	46	254	7	102	0	121310	4	4.0840			10
85	15	166	4	101	0	121903	2	4.1543	5		
86	25	213	5	101	0	121904	3	4.1739			11
87	71	317	6	151	3	121904	3	4.2412			12
88	70	316	6	151	0	121310	3	4.2563			13
89	69	312	6	51	0	121310	3	4.2597			14
90	57	278	16	52	-4	121310	0	4.2644	10		
91	61	289	6	101	3	121310	3	4.2899			15
92	68	307	6	21	3	121904	3	4.2984			16
93	48	264	6	102	3	121310	3	4.3640			17
94	42	246	7	101	0	121310	4	4.3905			11
95	107	262	6	102	0	121903	3	4.4200			18
96	13	1645	4	101	0	121903	2	4.5294		6	
97	17	169	4	101	-3	121310	2	4.6420		7	
98	103	169	4	101	-3	121903	2	4.7207		8	
99	52	273	16	52	2	121310	0	4.7743	11		
100	110	273	16	52	2	121903	0	5.0932	12		
101	119	262	6	102	0	121904	3	5.1485			19
102	47	262	6	102	0	121310	3	5.1505			20
103	66	304	16	51	-4	121904	0	5.6748	13		
104	40	243	12	101	0	121310	0	6.1170	14		
105	54	275	16	52	4	121904	0	6.7705	15		
106	56	277	16	52	-2	121310	0	6.8241	16		
107	63	296	16	51	0	121310	0	7.0360	17		
108	109	271	16	52	0	121903	0	7.0855	18		
109	121	274	16	52	4	121904	0	7.4104	19		
110	120	271	16	52	0	121904	0	7.5629	20		
111	55	276	16	52	-2	121310	0	7.6973	21		
112	49	270	16	52	0	121310	0	7.8023	22		
113	108	270	16	52	0	121903	0	7.9527	23		
114	50	271	16	52	0	121310	0	8.0455	24		
115	51	272	16	52	2	121310	0	8.2939	25		
116	33	224	12	101	0	121310	0	11.6656	26		
117	59	281	16	52	0	121310	0	15.0124	27		
118	60	283	16	52	-2	121310	0	15.5583	28		
119	58	280	16	52	0	121310	0	15.7680	29		
120	104	0	10	0	0	121903	5	20.0684			1
121	1	0	10	0	0	121903	5	101.7179			2

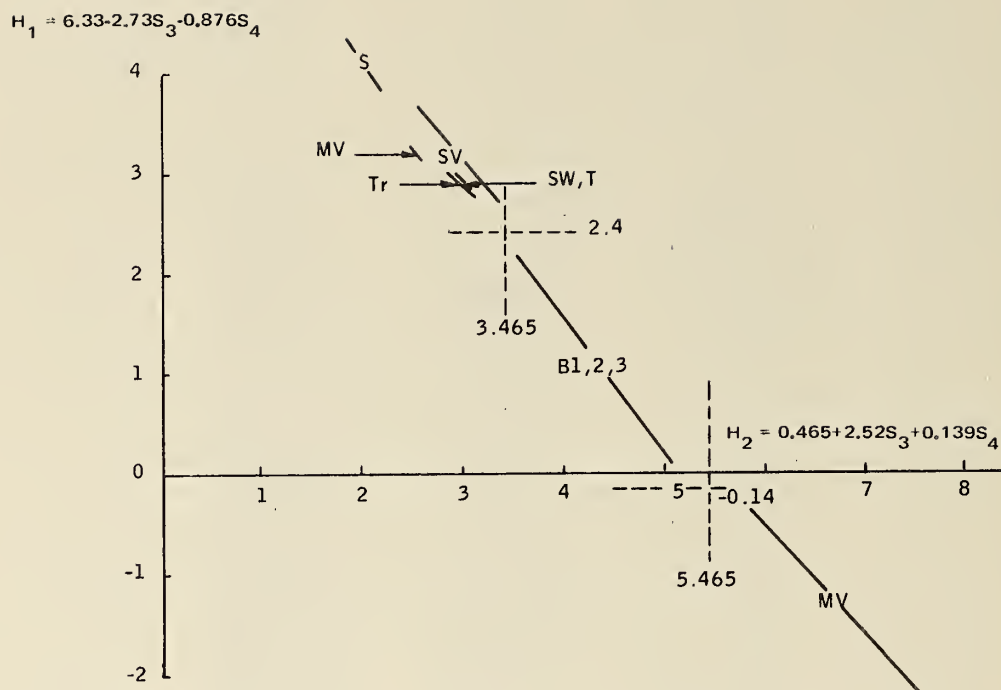


Figure 27. Joint Plot of Stopped Vehicle Discriminant Values

The stopped vehicle discriminants were found using the CTC program. A run was made using the method 1 classifier on secondary features three and four. Five classes were assigned vehicle types as follows:

- Class 1: Types 1, 2, 3 (3 bus types) (moving)
- Class 2: Type 4 (semi-trailer) (moving)
- Class 3: Type 5 (truck) (moving)
- Class 4: Type 6 (step van) (moving)
- Class 5: Type 7 (moving van) (moving)
- Class 0: All others (moving and stopped)

Each class was assigned an a-priori probability of 0.20, except class 0 which is assigned zero probability. The method 1 classifier calculated discriminants for all class pairs; the tradeoff of scores for the class 1 versus class 3 discriminant is shown in Table 43. Two thresholds along this discriminant were selected, one at a score value of -0.1408 and another at a score value of 2.399. These thresholds were chosen because all class 1 feature vectors

TABLE 43. TRADEOFF LISTING FOR BUS CLASSIFIER  
DESIGN

DATE 1223 ID BD RUN28  
TRADEOFF FOR  
PAIR 1 3

COEFF									
6.32919		-.00273		-.00088					
	NB	ID	TY		CL	SCORE	0	1	2 3 4 5
1	64	298	16 51	2 121310	0	-10.0128	1		
2	46	254	7 102	0 121310	5	-2.0326			1
3	97	246	7 101	0 121310	5	-1.7768			2
4	98	247	7 101	0 121310	5	-1.3142			3
5	44	250	7 101	6 121310	5	-.8579			4
6	45	251	7 101	6 121904	5	-.3781			5
7	82	344	3 151	3 121308	1	.0929		1	
8	32	222	2 101	-3 121310	1	.2604		2	
9	94	357	3 51	-3 121308	1	.2643		3	
10	33	224	12 101	0 121310	0	.3686	2		
11	39	241	2 101	-3 121310	1	.4885		4	
12	90	352	3 51	4 121308	1	.5084		5	
13	79	341	3 101	3 121308	1	.5623		6	
14	74	336	3 51	3 121308	1	.6561		7	
15	84	346	3 51	0 121308	1	.6563		8	
16	89	351	3 51	2 121308	1	.7064		9	
17	92	355	3 51	0 121308	1	.7788		10	
18	72	334	3 51	0 121308	1	.8163		11	
19	75	337	3 51	6 121308	1	.8280		12	
20	81	343	3 151	0 121308	1	.8702		13	
21	88	350	3 51	2 121308	1	.9748		14	
22	27	217	2 101	0 121310	1	.9802		15	
23	86	348	3 51	0 121308	1	.9858		16	
24	78	340	3 101	0 121308	1	.9983		17	
25	73	335	3 51	0 121308	1	1.0001		18	
26	80	342	3 101	-3 121308	1	1.0090		19	
27	76	338	3 51	-3 121308	1	1.0134		20	
28	85	347	3 51	0 121308	1	1.0893		21	
29	26	216	2 101	0 121310	1	1.0906		22	
30	83	345	3 151	-3 121308	1	1.1091		23	
31	2	99	1 101	0 121310	1	1.2003		24	
32	99	99	1 101	0 121903	1	1.2030		25	
33	87	349	3 51	0 121308	1	1.2186		26	
34	38	240	2 101	6 121310	1	1.3085		27	
35	95	358	13 51	0 121308	0	1.3247	3		
36	31	221	2 101	6 121310	1	1.3340		28	
37	5	101	1 101	0 121310	1	1.3473		29	
38	93	356	3 51	3 121308	1	1.3600		30	
39	91	353	3 51	-2 121308	1	1.3732		31	
40	96	359	13 51	0 121308	0	1.4480	4		
41	6	102	1 101	3 121310	1	1.4495		32	
42	35	237	2 101	0 121310	1	1.4659		33	
43	28	218	2 101	0 121310	1	1.4918		34	
44	29	219	2 101	0 121310	1	1.5088		35	
45	36	238	2 101	0 121310	1	1.5196		36	
46	100	995	1 101	0 121903	1	1.5502		37	
47	7	104	1 101	-3 121310	1	1.5600		38	
48	3	995	1 101	0 121310	1	1.5638		39	
49	4	100	1 101	0 121310	1	1.6006		40	
50	77	339	3 51	-6 121308	1	1.6159		41	
51	37	239	2 101	3 121310	1	1.7848		42	
52	30	220	2 101	3 121310	1	1.8016		43	
53	8	105	1 101	-6 121310	1	1.9137		44	



TABLE 43. TRADEOFF LISTING FOR BUS CLASSIFIER  
DESIGN (CONCLUDED)

54	34	225	12	101	0	121310	0	1.9268	5		
55	40	243	12	101	0	121310	0	2.1142	6		
56	58	280	16	52	0	121310	0	2.6955	7		
57	67	305	16	51	-4	121904	0	2.7122	8		
58	59	281	16	52	0	121310	0	2.7181	9		
59	66	304	16	51	-4	121904	0	2.7325	10		
60	10	149	9	101	0	121904	0	2.7340	11		
61	65	300	16	51	4	121904	0	2.7368	12		
62	25	213	5	101	0	121904	3	2.7383			
63	24	212	5	101	-6	121310	3	2.7544		1	
64	55	276	16	52	-2	121310	0	2.7763	13	2	
65	9	148	9	101	0	121903	0	2.7775	14		
66	47	262	6	102	0	121310	4	2.7910			1
67	52	273	16	52	2	121310	0	2.7990	15		
68	62	295	16	51	0	121310	0	2.8042	16		
69	56	277	16	52	-2	121310	0	2.8140	17		
70	51	272	16	52	2	121310	0	2.8349	18		
71	21	206	5	101	0	121903	3	2.8960		3	
72	11	157	8	101	0	121903	0	2.9224	19		
73	20	205	5	101	0	121310	3	2.9400		4	
74	12	158	8	101	0	121903	0	2.9546	20		
75	60	283	16	52	-2	121310	0	2.9768	21		
76	22	207	5	101	0	121310	3	2.9985		5	
77	63	296	16	51	0	121310	0	3.0000	22		
78	23	208	5	101	0	121310	3	3.0032		6	
79	71	317	6	151	3	121904	4	3.0805			2
80	69	312	6	51	0	121310	4	3.0922			3
81	70	316	6	151	0	121310	4	3.0958			4
82	50	271	16	52	0	121310	0	3.1199	23		
83	53	274	16	52	4	121310	0	3.1321	24		
84	54	275	16	52	4	121904	0	3.1350	25		
85	61	289	6	101	3	121310	4	3.1407		5	
86	68	307	6	21	3	121904	4	3.1453		6	
87	43	247	7	101	0	121310	5	3.1546			6
88	41	245	7	101	0	121904	5	3.1549			7
89	42	246	7	101	0	121310	5	3.2018			8
90	57	278	16	52	-4	121310	0	3.2090	26		
91	1	0	10	0	0	121903	0	3.2416	27		
92	48	264	6	102	3	121310	4	3.2984		7	
93	49	270	16	52	0	121310	0	3.5542	28		
94	13	1645	4	101	0	121903	2	3.8761			1
95	16	167	4	101	3	121310	2	3.9002			2
96	19	171	4	101	-6	121310	2	4.0106			3
97	14	165	4	101	0	121903	2	4.0158			4
98	18	170	4	101	-4	121310	2	4.0324			5
99	15	166	4	101	0	121903	2	4.1382			6
100	17	169	4	101	-3	121310	2	4.2809			7



lie between them, and all other feature vectors lie outside them (except for several class 0 vectors, which are of types 12 and 13, namely stopped busses). A logical variable was defined as true for scores greater than 2.399 and false otherwise, and a second logical variable was defined as true for scores greater than -0.1408 and false otherwise. These two logical variables could be used alone since they discriminate the range of scores associated with class 1 (buses), but two more logical variables were defined instead as additional selection criteria.

These last two logical variables were selected from the discriminant between class 2 and class 3, shown in Table 44. While it may appear thoughtless to use the boundary (discriminant) between classes 2 and 3 (semis and trucks, respectively) to detect a class 1 vehicle (bus), examining the scores reveals that all class 1 vectors lie between two limits - 3.465 and 5.465. Also the only other vectors in this range are from class 0 and are, in fact, non-moving busses. The two thresholds were used and logical variables were defined - one being true only for scores greater than 3.465 and the other being true for scores greater than 5.465. When the first is true and the second is false, then a bus is detected.

#### DECIDING WHETHER A VEHICLE IS STOPPED OR MOVING

The decision rules for stopped and for moving vehicles could be determined independent of the method used to decide whether a vehicle was a stopped vehicle or a moving vehicle. However, the decision has to be based upon the primary or secondary features if it is to be included in the bus classifier algorithm.

If a vehicle stops while crossing the loop, then one or more of the primary time features  $T_0$ ,  $T_1$ ,  $T_S$ ,  $T_L$  will become large (see Figure 16). The

TABLE 44. TRADEOFF LISTING FOR BUS CLASSIFIER  
DESIGN

DATE 1223 ID 50 RUN2E  
TRADEOFF FOR  
PAIR 2 3

CBEFF		.46471		.00252		.00014		CL	SCORE	0	1	2	3	4	5
		N8		ID TY											
1	17	169	4	101	-3	121310	2		1.8988						
2	18	170	4	101	-4	121310	2		1.9717			1			
3	15	166	4	101	0	121903	2		1.9758			2			
4	19	171	4	101	-6	121310	2		1.9919			3			
5	14	165	4	101	0	121903	2		2.0286			4			
6	16	167	4	101	3	121310	2		2.0658			5			
7	13	1645	4	101	0	121903	2		2.1886			6			
8	42	246	7	101	0	121310	5		2.5379			7			1
9	43	247	7	101	0	121310	5		2.5809						2
10	41	245	7	101	0	121904	5		2.5934						3
11	49	270	16	52	0	121310	0		2.6365	1					
12	57	278	16	52	-4	121310	0		2.7707	2					
13	48	264	6	102	3	121310	4		2.8253					1	
14	1	0	10	0	0	121903	0		2.8699	3					
15	23	208	5	101	0	121310	3		2.8764				1		
16	22	207	5	101	0	121310	3		2.8834				2		
17	20	205	5	101	0	121310	3		2.9261				3		
18	12	158	8	101	0	121903	0		2.9301	4					
19	68	307	6	21	3	121904	4		2.9395					2	
20	61	289	6	101	3	121310	4		2.9444					3	
21	11	157	8	101	0	121903	0		2.9686	5					
22	70	316	6	151	0	121310	4		2.9766					4	
23	53	274	16	52	4	121310	0		2.9791	6					
24	69	312	6	51	0	121310	4		2.9792					5	
25	71	317	6	151	3	121904	4		2.9873					6	
26	21	206	5	101	0	121903	3		2.9937				4		
27	54	275	16	52	4	121904	0		3.0100	7					
28	63	296	16	51	0	121310	0		3.0544	8					
29	66	304	16	51	-4	121904	0		3.0698	9					
30	67	305	16	51	-4	121904	0		3.0772	10					
31	24	212	5	101	-6	121310	3		3.0851				5		
32	9	148	9	101	0	121903	0		3.0918	11					
33	60	283	16	52	-2	121310	0		3.1207	12					
34	65	300	16	51	4	121904	0		3.1254	13					
35	25	213	5	101	0	121904	3		3.1315				6		
36	10	149	9	101	0	121904	0		3.1468	14					
37	62	295	16	51	0	121310	0		3.2421	15					
38	51	272	16	52	2	121310	0		3.2539	16					
39	56	277	16	52	-2	121310	0		3.2552	17					
40	50	271	16	52	0	121310	0		3.2713	18					
41	52	273	16	52	2	121310	0		3.2784	19					
42	55	276	16	52	-2	121310	0		3.2799	20					
43	47	262	6	102	0	121310	4		3.2818					7	
44	59	281	16	52	0	121310	0		3.3539	21					
45	58	280	16	52	0	121310	0		3.3721	22					
46	40	243	12	101	0	121310	0		3.5563	23					
47	8	105	1	101	-6	121310	1		3.5715		1				
48	77	339	3	51	-6	121308	1		3.7127		2				
49	34	225	12	101	0	121310	0		3.7156	24					
50	30	220	2	101	3	121310	1		3.7709		3				
51	37	239	2	101	3	121310	1		3.7778		4				
52	91	353	3	51	-2	121308	1		3.8286		5				
53	93	356	3	51	3	121308	1		3.8870		6				

TABLE 44. TRADEOFF LISTING FOR BUS CLASSIFIER  
DESIGN (CONCLUDED)

54	4	100	1	101	0	121310	1	3.8885	7
55	96	359	13	51	0	121308	0	3.9002	25
56	3	995	1	101	0	121310	1	3.9131	8
57	7	104	1	101	-3	121310	1	3.9200	9
58	31	221	2	101	6	121310	1	3.9225	10
59	100	995	1	101	0	121903	1	3.9236	11
60	38	240	2	101	6	121310	1	3.9453	12
61	95	358	13	51	0	121308	0	3.9887	26
62	6	102	1	101	3	121310	1	3.9960	13
63	36	238	2	101	0	121310	1	4.0559	14
64	28	218	2	101	0	121310	1	4.0561	15
65	29	219	2	101	0	121310	1	4.0638	16
66	5	101	1	101	0	121310	1	4.0790	17
67	35	237	2	101	0	121310	1	4.0895	18
68	87	349	3	51	0	121308	1	4.1116	19
69	83	345	3	151	-3	121308	1	4.1190	20
70	99	99	1	101	0	121903	1	4.1730	21
71	2	99	1	101	0	121310	1	4.1755	22
72	76	338	3	51	-3	121308	1	4.1780	23
73	80	342	3	101	-3	121308	1	4.1787	24
74	85	347	3	51	0	121308	1	4.1889	25
75	75	337	3	51	6	121308	1	4.2033	26
76	88	350	3	51	2	121308	1	4.2405	27
77	73	335	3	51	0	121308	1	4.2553	28
78	78	340	3	101	0	121308	1	4.2577	29
79	86	348	3	51	0	121308	1	4.2826	30
80	81	343	3	151	0	121308	1	4.3386	31
81	72	334	3	51	0	121308	1	4.3931	32
82	89	351	3	51	2	121308	1	4.4210	33
83	92	355	3	51	0	121308	1	4.4324	34
84	84	346	3	51	0	121308	1	4.4331	35
85	26	216	2	101	0	121310	1	4.4351	36
86	74	336	3	51	3	121308	1	4.4561	37
87	90	352	3	51	4	121308	1	4.4566	38
88	79	341	3	101	3	121308	1	4.5186	39
89	27	217	2	101	0	121310	1	4.5090	40
90	94	357	3	51	-3	121308	1	4.6562	41
91	82	344	3	151	3	121308	1	4.7878	42
92	39	241	2	101	-3	121310	1	4.8920	43
93	33	224	12	101	0	121310	0	4.9653	27
94	32	222	2	101	-3	121310	1	5.0660	44
95	45	251	7	101	6	121904	5	5.8481	
96	44	250	7	101	6	121310	5	6.1081	4
97	98	247	7	101	0	121310	5	6.7130	5
98	97	246	7	101	0	121310	5	7.1415	6
99	46	254	7	102	0	121310	5	7.4034	7
100	64	298	16	51	2	121310	0	13.2835	28

normalized time features  $E_1 = \frac{T_0}{T_0 + T_L}$  and  $E_2 = \frac{T_1 + T_S}{T_0 + T_L}$  will behave in distinctive ways that depend on which of the times  $T_i$  is large. Thus, if  $T_L$  becomes very large, then both  $E_1$  and  $E_2$  will become small. The other possibilities are:

<u>Large Time Value</u>	<u><math>E_1</math></u>	<u><math>E_2</math></u>
$T_0$	Large	Small
$T_1$	---	Large
$T_S$	---	Large
$T_L$	Small	Small

Thus, we should expect moving vehicles to have small values of  $E_2$  and intermediate values for  $E_1$ .

Study of the distribution of  $E_1$  and  $E_2$  for the experimental data led to defining small  $E_2$  as  $E_2 < 2.4$  and intermediate  $E_1$  as  $0.43 \leq E_1 < 0.70$ .

To determine a decision rule based on the experimental data, define two logical variables  $L_1$ ,  $L_2$  where

$L_1$  is true when  $0.43 \leq E_1 < 0.70$

$L_2$  is true when  $2.5 \leq E_2$ .

The values (true or false) of  $L_1$  and  $L_2$  determined from the test data are:

$L_1 =$	T	T	F	F
$L_2 =$	T	F	F	T
Number of Moving Vehicles	7	79	3	0
Number of Stopped Vehicles	16	2	8	0



Thus, it is clear that  $L_1$  true and  $L_2$  false is the condition to use for deciding that a vehicle was moving.

It was found that certain moving vehicles were more easily classified by the stopped vehicle classifier. Study of the signal recordings and logs revealed that these vehicles had been accelerating or decelerating as they crossed the loop. The thresholds were then adjusted and the two final vehicle divisions are more appropriately called uniformly and nonuniformly moving vehicles.

### TESTING THE COMPLETE CLASSIFIER

The output obtained using the logical mode classifier with the two logical variables to classify moving vehicles and the three logical variables to distinguish between uniformly moving and non-uniformly moving vehicles discussed above is shown in Table 45. All moving busses (class 1) have scores equal to 20 (octal), and all other vehicles have scores not equal to 20. Thus a bus is detected for any feature vector input to the classifier which produces an output score equal to 20.

A final run was made using the logical mode classifier with all nine logical variables. Five of these were the same as those used in the previously discussed logical mode execution, and the other four were those determined from discriminant scoring values obtained from the method 1 classifier as classifying non-uniformly moving vehicles. These nine logical variables were used in an execution of the logical mode classifier; the output is shown in Table 46. In this case there are two scores which must be detected to indicate a bus. A score of 400 has been assigned to one event, while all other busses are scored at 412. Table 47 summarizes the results obtained using the CTC program. Figure 28 is a functional diagram of the complete bus detector from sensor (the buried loop) to final classification (bus or not bus).



TABLE 45. TRADEOFF LISTING FOR STOPPED VEHICLE CLASSIFIER

DATE 624 ID RJN BD1									
LCOUNT = 9									
	NO	ID	TY		CL	SCORE	0	1	2 3 4
1	68	205	16	51	-4 121904	4	00000000		1
2	145	271	16	52	0 121904	4	00000060		2
3	135	273	16	52	2 121903	4	00000060		3
4	134	271	16	52	0 121903	4	00000060		4
5	53	273	16	52	2 121310	4	00000060		5
6	51	271	16	52	0 121310	4	00000060		6
7	61	283	16	52	-2 121310	4	00000060		7
8	58	278	16	52	-4 121310	4	00000060		8
9	127	101	1	101	0 121903	1	00000400	1	
10	140	220	2	101	3 121904	1	00000412	2	
11	139	219	2	101	0 121904	1	00000412	3	
12	138	218	2	101	0 121904	1	00000412	4	
13	137	217	2	101	0 121904	1	00000412	5	
14	136	216	2	101	0 121904	1	00000412	6	
15	126	100	1	101	0 121903	1	00000412	7	
16	125	995	1	101	0 121903	1	00000412	8	
17	124	99	1	101	0 121903	1	00000412	9	
18	95	357	3	51	-3 121308	1	00000412	10	
19	94	356	3	51	3 121308	1	00000412	11	
20	93	355	3	51	0 121308	1	00000412	12	
21	92	353	3	51	-2 121308	1	00000412	13	
22	91	352	3	51	4 121308	1	00000412	14	
23	90	351	3	51	2 121308	1	00000412	15	
24	89	350	3	51	2 121308	1	00000412	16	
25	88	349	3	51	0 121308	1	00000412	17	
26	87	348	3	51	0 121308	1	00000412	18	
27	86	347	3	51	0 121308	1	00000412	19	
28	85	346	3	51	0 121308	1	00000412	20	
29	84	345	3	151	-3 121308	1	00000412	21	
30	83	344	3	151	3 121308	1	00000412	22	
31	82	343	3	151	0 121308	1	00000412	23	
32	81	342	3	101	-3 121308	1	00000412	24	
33	80	341	3	101	3 121308	1	00000412	25	
34	79	340	3	101	0 121308	1	00000412	26	
35	78	339	3	51	-6 121308	1	00000412	27	
36	77	338	3	51	-3 121308	1	00000412	28	
37	76	337	3	51	6 121308	1	00000412	29	
38	75	336	3	51	3 121308	1	00000412	30	
39	74	335	3	51	0 121308	1	00000412	31	
40	73	334	3	51	0 121308	1	00000412	32	
41	40	241	2	101	-3 121310	1	00000412	33	
42	39	240	2	101	6 121310	1	00000412	34	
43	38	239	2	101	3 121310	1	00000412	35	
44	37	238	2	101	0 121310	1	00000412	36	
45	36	237	2	101	0 121310	1	00000412	37	
46	33	222	2	101	-3 121310	1	00000412	38	
47	32	221	2	101	6 121310	1	00000412	39	
48	31	220	2	101	3 121310	1	00000412	40	
49	30	219	2	101	0 121310	1	00000412	41	
50	29	218	2	101	0 121310	1	00000412	42	
51	28	217	2	101	0 121310	1	00000412	43	
52	27	216	2	101	0 121310	1	00000412	44	
53	9	105	1	101	-6 121310	1	00000412	45	
54	8	104	1	101	-3 121310	1	00000412	46	
55	6	102	1	101	3 121310	1	00000412	47	
56	5	101	1	101	0 121310	1	00000412	48	
57	4	100	1	101	0 121310	1	00000412	49	
58	3	995	1	101	0 121310	1	00000412	50	

TABLE 45. TRADEOFF LISTING FOR STOPPED VEHICLE  
CLASSIFIER (CONCLUDED)

59	2	99	1	101	0	121310	1	00000020	51	
60	44	247	7	101	0	121310	3	00000021		1
61	130	205	5	101	0	121903	3	00000022		2
62	25	212	5	101	-6	121310	3	00000022		3
63	24	208	5	101	0	121310	3	00000022		4
64	23	207	5	101	0	121310	3	00000022		5
65	21	205	5	101	0	121310	3	00000022		6
66	13	158	8	101	0	121903	3	00000022		7
67	12	157	8	101	0	121903	3	00000022		8
68	10	148	9	101	0	121903	3	00000022		9
69	144	252	6	102	0	121904	3	00000023		10
70	142	246	7	101	0	121904	3	00000023		11
71	132	252	6	102	0	121903	3	00000023		12
72	128	169	4	101	-3	121903	3	00000023		13
73	123	247	7	101	0	121310	3	00000023		14
74	72	317	6	151	3	121904	3	00000023		15
75	71	316	6	151	0	121310	3	00000023		16
76	70	312	6	51	0	121310	3	00000023		17
77	69	307	6	21	3	121904	3	00000023		18
78	62	289	6	101	3	121310	3	00000023		19
79	57	277	16	52	-2	121310	4	00000023		20
80	55	275	16	52	4	121904	4	00000023		10
81	49	254	6	102	3	121310	3	00000023		20
82	48	262	6	102	0	121310	3	00000023		21
83	42	245	7	101	0	121904	3	00000023		22
84	26	213	5	101	0	121904	3	00000023		23
85	22	206	5	101	0	121903	3	00000023		24
86	18	159	4	101	-3	121310	3	00000023		25
87	16	156	4	101	0	121903	3	00000023		26
88	15	165	4	101	0	121903	3	00000023		27
89	14	1645	4	101	0	121903	3	00000023		28
90	11	149	9	101	0	121904	3	00000023		29
91	147	298	16	51	2	121904	4	00000024		11
92	141	225	12	101	0	121904	2	00000024	1	
93	97	359	13	51	0	121308	2	00000024	2	
94	65	298	16	51	2	121310	4	00000024		12
95	35	225	12	101	0	121310	2	00000024	3	
96	122	246	7	101	0	121310	3	00000025		30
97	96	358	13	51	0	121308	2	00000025	4	
98	67	304	16	51	-4	121904	4	00000025		13
99	43	246	7	101	0	121310	3	00000025		31
100	41	243	12	101	0	121310	2	00000025	5	
101	34	224	12	101	0	121310	2	00000025	6	
102	146	274	16	52	4	121904	4	00000027		14
103	133	270	16	52	0	121903	4	00000027		15
104	131	254	7	102	0	121903	3	00000027		32
105	60	251	16	52	0	121310	4	00000027		16
106	59	230	16	52	0	121310	4	00000027		17
107	56	276	16	52	-2	121310	4	00000027		18
108	52	272	16	52	2	121310	4	00000027		19
109	50	270	16	52	0	121310	4	00000027		20
110	47	254	7	102	0	121310	3	00000027		33
111	20	171	4	101	-6	121310	3	00000027		34
112	19	170	4	101	-4	121310	3	00000027		35
113	17	167	4	101	3	121310	3	00000027		36
114	1	0	0	0	0	121903	0	00000027	1	
115	45	250	7	101	6	121310	3	00000031		37
116	143	250	7	101	6	121904	3	00000033		38
117	66	300	16	51	4	121904	4	00000033		21
118	63	295	16	51	0	121310	4	00000033		22
119	54	274	16	52	4	121310	4	00000033		23
120	46	251	7	101	6	121904	3	00000033		39
121	129	0	0	0	0	121903	0	00000037	2	
122	64	296	16	51	0	121310	4	00000037		24

INPUT

TABLE 46. TRADEOFF LISTING FOR COMPLETE CLASSIFIER

DATE 624 ID RUN BD2												
LCOUNT = 5												
N0	ID	TY				CL	SCORE	0	1	2	3	4
1	68	305	16	51	-4	121904	4	00000000				1
2	145	271	16	52	0	121904	4	00000003				2
3	135	273	16	52	2	121903	4	00000003				3
4	134	271	16	52	0	121903	4	00000003				4
5	53	273	16	52	2	121310	4	00000003				5
6	51	271	16	52	0	121310	4	00000003				6
7	61	283	16	52	-2	121310	4	00000007				7
8	58	278	16	52	-4	121310	4	00000007				8
9	140	220	2	101	3	121904	1	00000020	1			
10	139	219	2	101	0	121904	1	00000020	2			
11	138	218	2	101	0	121904	1	00000020	3			
12	137	217	2	101	0	121904	1	00000020	4			
13	136	216	2	101	0	121904	1	00000020	5			
14	127	101	1	101	0	121903	1	00000020	6			
15	126	100	1	101	0	121903	1	00000020	7			
16	125	995	1	101	0	121903	1	00000020	8			
17	124	99	1	101	0	121903	1	00000020	9			
18	95	357	3	51	-3	121308	1	00000020	10			
19	94	356	3	51	3	121308	1	00000020	11			
20	93	355	3	51	0	121308	1	00000020	12			
21	92	353	3	51	-2	121308	1	00000020	13			
22	91	352	3	51	4	121308	1	00000020	14			
23	90	351	3	51	2	121308	1	00000020	15			
24	89	350	3	51	2	121308	1	00000020	16			
25	88	349	3	51	0	121308	1	00000020	17			
26	87	348	3	51	0	121308	1	00000020	18			
27	86	347	3	51	0	121308	1	00000020	19			
28	85	346	3	51	0	121308	1	00000020	20			
29	84	345	3	151	-3	121308	1	00000020	21			
30	83	344	3	151	3	121308	1	00000020	22			
31	82	343	3	151	0	121308	1	00000020	23			
32	81	342	3	101	-3	121308	1	00000020	24			
33	80	341	3	101	3	121308	1	00000020	25			
34	79	340	3	101	0	121308	1	00000020	26			
35	78	339	3	51	-6	121308	1	00000020	27			
36	77	338	3	51	-3	121308	1	00000020	28			
37	76	337	3	51	6	121308	1	00000020	29			
38	75	336	3	51	3	121308	1	00000020	30			
39	74	335	3	51	0	121308	1	00000020	31			
40	73	334	3	51	0	121308	1	00000020	32			
41	40	241	2	101	-3	121310	1	00000020	33			
42	39	240	2	101	6	121310	1	00000020	34			
43	38	239	2	101	3	121310	1	00000020	35			
44	37	238	2	101	0	121310	1	00000020	36			
45	36	237	2	101	0	121310	1	00000020	37			
46	33	222	2	101	-3	121310	1	00000020	38			
47	32	221	2	101	6	121310	1	00000020	39			
48	31	220	2	101	3	121310	1	00000020	40			
49	30	219	2	101	0	121310	1	00000020	41			
50	29	218	2	101	0	121310	1	00000020	42			
51	28	217	2	101	0	121310	1	00000020	43			
52	27	216	2	101	0	121310	1	00000020	44			
53	9	105	1	101	-6	121310	1	00000020	45			
54	8	104	1	101	-3	121310	1	00000020	46			
55	6	102	1	101	3	121310	1	00000020	47			
56	5	101	1	101	0	121310	1	00000020	48			
57	4	100	1	101	0	121310	1	00000020	49			
58	3	995	1	101	0	121310	1	00000020	50			



TABLE 46. TRADEOFF LISTING FOR COMPLETE CLASSIFIER (CONCLUDED)

59	2	99	1	101	0	121310	1	00000412	51	
60	44	247	7	101	0	121310	3	00000420		1
61	130	205	5	101	0	121903	3	00000440		2
62	25	212	5	101	-6	121310	3	00000440		3
63	24	208	5	101	0	121310	3	00000440		4
64	23	207	5	101	0	121310	3	00000440		5
65	21	205	5	101	0	121310	3	00000440		6
66	13	158	8	101	0	121903	3	00000440		7
67	12	157	8	101	0	121903	3	00000440		8
68	10	148	9	101	0	121903	3	00000440		9
69	144	262	6	102	0	121904	3	00000460		10
70	142	246	7	101	0	121904	3	00000460		11
71	132	262	6	102	0	121903	3	00000460		12
72	128	169	4	101	-3	121903	3	00000460		13
73	72	317	6	151	3	121904	3	00000460		14
74	71	316	6	151	0	121310	3	00000460		15
75	70	312	6	51	0	121310	3	00000460		16
76	69	307	6	21	3	121904	3	00000460		17
77	62	289	6	101	3	121310	3	00000460		18
78	57	277	16	52	-2	121310	4	00000460		9
79	55	275	16	52	4	121904	4	00000460		10
80	49	264	6	102	3	121310	3	00000460		19
81	48	262	6	102	0	121310	3	00000460		20
82	42	245	7	101	0	121904	3	00000460		21
83	26	213	5	101	0	121904	3	00000460		22
84	22	206	5	101	0	121903	3	00000460		23
85	18	169	4	101	-3	121310	3	00000460		24
86	16	166	4	101	0	121903	3	00000460		25
87	15	165	4	101	0	121903	3	00000460		26
88	14	1645	4	101	0	121903	3	00000460		27
89	11	149	9	101	0	121904	3	00000460		28
90	123	247	7	101	0	121310	3	00000477		29
91	141	225	12	101	0	121904	2	00000512		1
92	97	359	13	51	0	121308	2	00000512		2
93	35	225	12	101	0	121310	2	00000512		3
94	147	298	16	51	2	121904	4	00000516		11
95	65	298	16	51	2	121310	4	00000517		12
96	67	304	16	51	-4	121904	4	00000520		13
97	43	246	7	101	0	121310	3	00000520		30
98	96	358	13	51	0	121308	2	00000532		4
99	41	243	12	101	0	121310	2	00000532		5
100	34	224	12	101	0	121310	2	00000532		6
101	122	246	7	101	0	121310	3	00000537		31
102	146	274	16	52	4	121904	4	00000560		14
103	133	270	16	52	0	121903	4	00000560		15
104	60	281	16	52	0	121310	4	00000560		16
105	59	280	16	52	0	121310	4	00000560		17
106	56	276	16	52	-2	121310	4	00000560		18
107	52	272	16	52	2	121310	4	00000560		19
108	50	270	16	52	0	121310	4	00000560		20
109	20	171	4	101	-6	121310	3	00000560		32
110	19	170	4	101	-4	121310	3	00000560		33
111	17	167	4	101	3	121310	3	00000560		34
112	1	0	0	0	0	121903	0	00000560	1	
113	131	254	7	102	0	121903	3	00000577		35
114	47	254	7	102	0	121310	3	00000577		36
115	45	250	7	101	6	121310	3	00000637		37
116	66	300	16	51	4	121904	4	00000660		21
117	63	295	16	51	0	121310	4	00000660		22
118	54	274	16	52	4	121310	4	00000660		23
119	143	250	7	101	6	121904	3	00000677		28
120	46	251	7	101	6	121904	3	00000677		29
121	129	0	0	0	0	121903	0	00000760	2	
122	64	296	16	51	0	121310	4	00000760		24

INPUT

TABLE 47. BUS CLASSIFIER DECISION ALGORITHM

$L_1, \dots, H_2$  are the logical variables defined in the preceding discussions.

Stopped Vehicle Test

$L_1$  true when  $430 \leq S_6 < 700$

$L_2$  true when  $2500 \leq S_7$

A non-stopped vehicle is indicated when  $L_1$  is true and  $L_2$  is false.

Moving Vehicle Classifier

$K_1$  true when

$$-0.277 S_1 + 0.0198 S_2 + 0.228 S_3 - 0.947 S_4 \geq -1.01$$

$K_2$  true when

$$-1.66 S_1 + 1.36 S_2 + 2.17 S_3 - 4.10 S_4 \geq -1.944$$

A "bus" is indicated when  $K_1$  is false and  $K_2$  is false; otherwise a "non-bus" is indicated.

Stopped Vehicle Classifier

$H_1$  is true when

$$3.93 \leq 2.73 S_3 + 0.876 S_4 \leq 6.47$$

$H_2$  is true when

$$3.00 \leq 2.52 S_3 + 0.139 S_4 \leq 5.00$$

A "bus" is indicated when  $H_1$  is true and  $H_2$  is true; otherwise, the vehicle is a "non-bus".



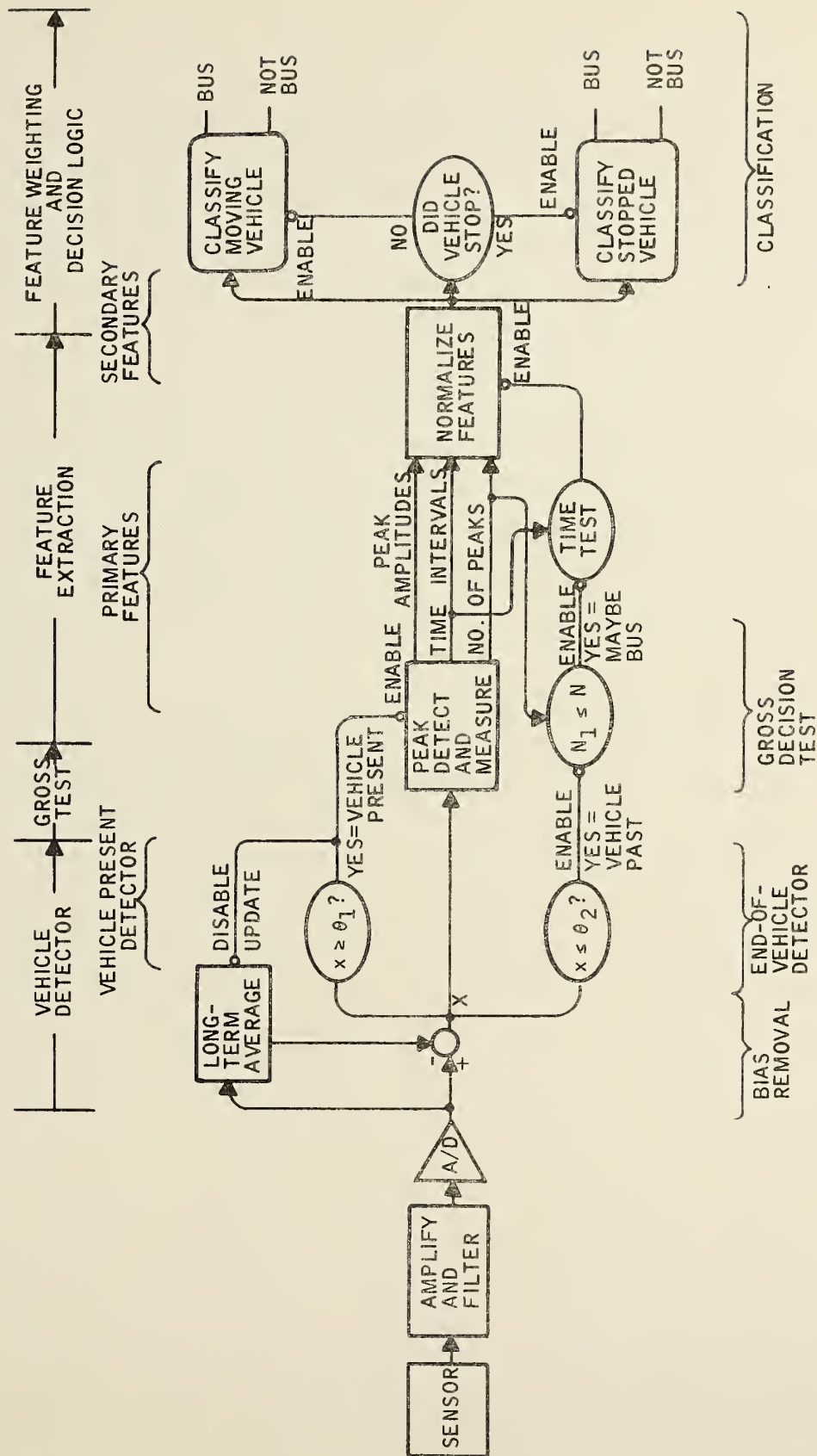


Figure 28. Functional Diagram of the Bus Detector

Subsequent to the initial classifier design, additional experimental data were obtained from the FHWA facility in Washington, D. C., plus recordings of street traffic in Washington, D. C. This data, plus the original data, were all processed, using the full classifier and, after minor changes to threshold values, the classifier correctly identified all signals with the exception of three questionable recordings.

### Areas of Potential Concern

Classifier Coefficients -- Geometrically speaking, the classifier coefficients describe the location and normal direction of planes that separate the bus feature values from the non-bus feature values. The feature values are calculated from measured peak times and amplitudes. These, in turn, depend upon the vehicle detection threshold, the peak tolerance value and the bias estimating filter. These were described in the preceding pages but there is a certain amount of arbitrariness in the whole procedure.

The threshold, tolerance, and bias estimate were defined and simulated and experiments were performed (using recorded signals) to determine good values for these parameters. Optimization was not possible because the optimum is the set that gives the best classification. Instead, the classifier is determined from the somewhat arbitrary choice of good values. Next, the features were defined by postulating a number of ways of combining measurements to overcome the stopped vehicle, different speeds, displacement from center line, and other variables that were apparent. Four of the candidate features were chosen because they had good statistical properties. Again, the choice was not a strict optimal one. Finally, the classifier coefficients were determined and the unpredictable result was perfect separation. This result produces one more arbitrary factor, because the slope of the planes can be changed slightly or they can be shifted back and forth a small distance and the classification is still 100 percent. Ideally, we should now

return to the starting point and optimize the detection threshold, the peak tolerance, and all the other arbitrarily chosen values so as to produce the best classifier. And as more data become available the loop should be recycled to further improve the results. In other words, the ideal design process does not end until the optimum is realized, which may be never.

Thus, we must conclude that some arbitrariness is unavoidable if we are to be practical. But we must also decide whether the current level of arbitrariness is acceptable. We can only answer on the basis of data, experience and intuition, and the answer at this time is "yes"! However, we may learn otherwise as our data and experience changes! Fortunately, the digital implementation allows such changes to be made relatively easily, so the decision to accept current level information is not as critical as for a hard-wired device.

Vehicle Detection Threshold -- The threshold was chosen before the classifier structure and the kinds of features were known. The primary criterion, at that time, was to detect all busses. A secondary criterion was to detect enough other vehicles so that a representative set of non-bus signatures could be obtained for the classifier training. Later on, the threshold became important in another way; it determines the first and last signal intervals, that in turn are used to normalize the time features to reduce their dependence on vehicle speed. After the classifier had been simulated, the threshold was tested by varying it over the 50 percent to 200 percent range to see if it was a critical value. Over this range, it was not, so the value was accepted.

Months later, when analyzing the Washington Street data with a simulation that now included the quantized values of time, amplitude and bias correction, it was seen that signal gain, quantization, bias correction and adjacent lane vehicles can couple with the threshold to produce errors. The effect can be counteracted for the existing data recordings by changing the stopped vehicle test from  $E_2 > 2.5$  to  $E_2 > 2.3$ . This corrects the one bus that experiences



this unique set of conditions and does not change the classification of any other vehicle. (It should be noted that this one bus has been run ten times on the simulation using different bias correction time constants and it was classified as a bus six times. The fundamental random variable appears to be the time of day at which the data is sampled!)

The change in the  $E_2$  test value is almost insignificant, but it shows a number of things. First, it is an example of the arbitrariness mentioned earlier. The data available did not require or allow a more accurate determination of this value. Second, the change in the  $E_2$  test value does cause some moving busses to be processed by what has been called the stop bus classifier. The busses that are affected are ones that are accelerating or decelerating. The "stopped bus classifier" thus becomes the "nonuniform velocity bus classifier". In retrospect this is much more reasonable - more data, better definition. The velocity estimate is not changed in any way by this. Third, we have an effect caused by the interaction of a number of "real-world" factors that was not seen when the parameters were varied one at a time in the simulation.

Fourth, it points out a way in which the chosen threshold value is very sub-optimal. This is discussed in the following paragraph.

Figure 29 illustrates the leading edge of a vehicle signature. The values  $M_1$ ,  $M_2$ ,  $M_3$  are amplitudes (phase values) above the zero or steady-state bias level and the  $T_i$  are the corresponding times.

If  $M_2$  were the chosen threshold value and the intervals between the  $M_1$  values is the amplitude quantization, then relatively large time errors are caused by the minimal ( $\pm$ ) quantization unit change in the bias estimate. Figure 30 shows how the time value is made insensitive to bias estimate errors simply by using a larger vehicle detection threshold that places the threshold in the linear, large slope part of the signal preceding the first peak.

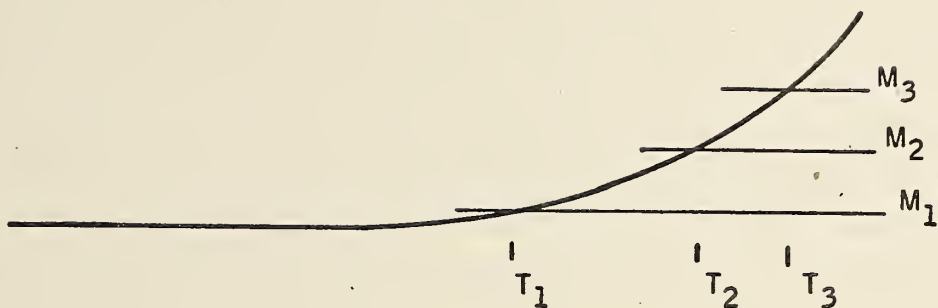


Figure 29. Leading Edge of a Vehicle Signature

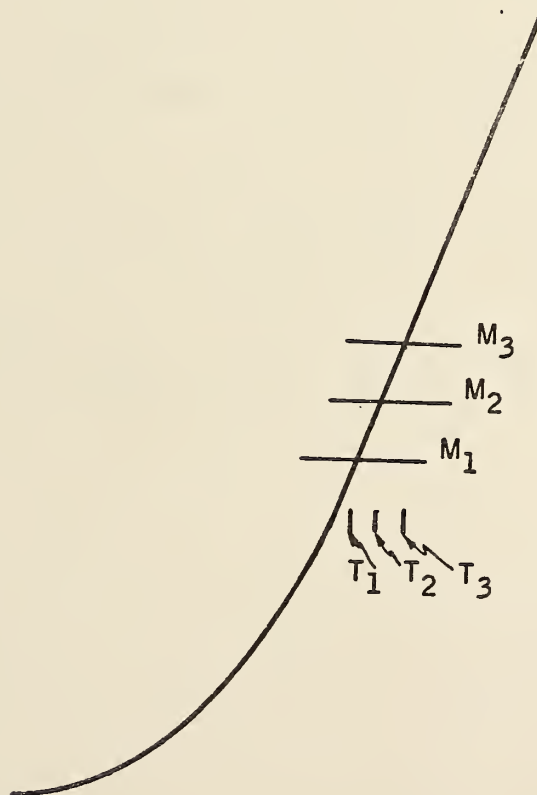


Figure 30. Magnitude versus Time



Similar coupling of the start (and stop) time exists with center line displacement, lead length, asphalt versus reinforced concrete, depth of paving material, loop age and any other factor that affects the signal peak phase value. Using a larger vehicle detection threshold would thus reduce the statistical variance of the vehicle classes and increase the confidence in the classifier. Unfortunately, changing the threshold changes the time-normalizing factor in a nonlinear manner and this affects the decision boundary planes; i. e., a completely new set of classifier coefficients is needed, but the structure is the same. Since all of this analysis is the result of one exceptional signal, there is not enough evidence to justify iterating the design loop at this time. But, if the problem becomes a large one, the following steps can be taken:

- 1) Run the recorded tapes on the simulation using larger detection threshold values and doing quantized arithmetic. List bias and all features.
- 2) Perform graphical analysis of  $T_i$ ,  $M_i$ ,  $E_i$ ,  $S_i$ ,  $L_i$ ,  $H_i$ ,  $K_i$  variation with the detection threshold.
- 3) Choose a value for the detection threshold that lies well within the linear part of the signal and which detects all busses.
- 4) If the graphical analysis in 2) shows a functional or highly correlated relationship between the threshold and the feature values, then use this to determine the new classifier coefficients. If the graphical analysis shows no simplifying relationship, then retrain the classifier using the new feature values.
- 5) Test the new classifier coefficients using recorded data tapes.

Bias Estimation -- When no vehicle is present the output of the loop phase detector is some nonzero constant value. This is the bias. The bias is expected to change slowly with time, temperature, etc., and it must be removed from the signal value before vehicle detection or feature extraction are done. The normalized amplitude features are insensitive to bias estimate errors, but the normalized time features are quite sensitive as discussed above. The sequel will summarize our experience with the bias and with ways of estimating it and then discuss some of the problems that may arise.

The expected total range of bias values is less than 8 deg (i. e. , if the loop were initially calibrated to output 0 deg with no vehicle present, then -4 deg to +4 deg will contain all future bias values). The initial simulation estimated the bias with a digital recursive low-pass filter that operated only when no vehicle was present (i. e. , signal minus bias less than detection threshold). Time constants in the range 1/10 sec - 10 sec were satisfactory. This algorithm involves a multiplication, which the preprocessor cannot perform. A second algorithm using shifts to replace the multiply was tested for the preprocessor implementation. This did not work because the 12-bit preprocessor word was too small. We next tried a method that adds 1 to the bias estimate when the nonvehicle signal is greater than the estimate and subtracts 1 when the signal is less than the estimate. This is about as simple as a low-pass filter can be, and, when tested, it worked.

The disadvantage of this up/down counter method of filtering lies in the interrelation between time constant and quantization. The time constant is set by the time interval between successive bias tests (the bias estimate is compared with the sampled signal value only once every N samples). A count of 1 up or down corresponds to the signal quantization value ( $\sim 1/4$  deg at present). Thus if the time interval is too small the filter may jump up and down rapidly, so the time interval is chosen large to make the output relatively smooth. But, since this filter does not smooth the values finer than

the quantization level, an error in the bias estimate cannot be corrected for the full time interval nor is it compensated for by smoothing. This causes a problem when the bias correction time happens to occur when the signal is increasing due to an approaching vehicle but before the signal reaches the threshold. In this case the bias will be increased one quantization interval above its correct value and the time estimates will be in error, as discussed previously.

For a large bias correction time interval (2-4 sec), this situation is infrequent since the increasing part of the signal is less than 0.2 sec long. But the error can occur another way. A vehicle in the adjacent lane or a lane straddler can cause a small increase in signal level that looks like a rapid bias increase. If this happens before a vehicle comes by in the main lane then the bias estimate is almost certain to be too large. Something of this happened in about 25 percent of the Washington street data. Using a larger vehicle detection threshold, as discussed in 2), would cure this problem by making the time features insensitive to bias errors. An alternate but less desirable way would be to introduce bias smoothing.

By using scaled values (stored value =  $2^8 \times$  bias estimate) the smallest bias estimate increment can be made a fraction of the quantization interval. Thus an error would have to persist for a number of correction intervals before its effect would appear in the lowest significant position. To implement this in the preprocessor without using the shift instructions:

```

Scaled bias estimate  $\longrightarrow$   $A_0$ 
add bias increment
 $A_0 \longrightarrow$  memory = new scaled bias estimate
 $A_0$  bits 8 - 11  $\longrightarrow$   $A_1$  = bias, unscaled

```

This would handle a 0 deg - 4 deg range of bias value, and could be expanded to 0 deg - 8 deg by using arithmetic overflow detection.



For the sake of simplicity of implementation and because the simulation was relatively insensitive to the following method of bias estimation, a simple scheme of bias correction was employed. At present, the preprocessor adds or subtracts 1 from the bias estimate if the estimate is less than or greater than the nonvehicle signature value. The estimate is not changed if it is equal to the nonvehicle signature value. This update process occurs once every 1.5 seconds.

## CHAPTER 7

### REFERENCES

- H. L. Van Trees, "Decision, Estimation and Modulation Theory," Vol. 1, John Wiley & Sons, 1968.
- G. F. Hughes, On the Mean Accuracy of Statistical Pattern Recognition, IEEE Transactions on Information Theory, Jan. 1968.
- K. Fukunaga, An Algorithm for finding Intrinsic Dimensionality of Data, IEEE Transactions on Electronic Computers, Feb. 1971.
- W. S. Meisel, "Computer-Oriented Approaches to Pattern Recognition," Academic Press, 1974.
- K. S. Fu, "Syntactic Methods in Pattern Recognition," Academic Press, 1974.



APPENDIX A  
DERIVATION OF LINEAR DISCRIMINANT  
CLASSIFIER FOR THE 2-CLASS UG CASE

A-priori -  $P_{C_j}$ ,  $j = 1, 2$

$$\text{density } P(X|C_j) = [(2\pi)^n \det V_{X,j}]^{-1/2} \exp[-\frac{1}{2} (X-m_j)^T V_{X,j}^{-1} (X-m_j)]$$

MAXIMUM LIKELIHOOD DECISION RULE DERIVATION  
(MINIMUM PROBABILITY OF ERROR)

Suppose classification rule is:

$$d(X) = \begin{cases} C_1 & : X \in X_F(1) \\ C_2 & : X \in X_F(2) \end{cases} \quad (A-1)$$

where

$$X_F(1) \cup X_F(2) = X_F, \quad X_F(1) \cap X_F(2) = \phi$$

Then

$$\begin{aligned} P_E &= P_{C_2} \Pr [d(X) = C_1 | C_2] + P_{C_1} \Pr [d(X) = C_2 | C_1] \\ &= P_{C_2} \int_{X_F(1)} P(X|C_2) dX + P_{C_1} \int_{X_F(2)} P(X|C_1) dX, \end{aligned} \quad (A-2)$$

or since

$$X_F(2) = X_F - X_F(1)$$

$$\begin{aligned} P_E &= P_{C_2} \int_{X_F(1)} P(X|C_2) dX + P_{C_1} \int_{X_F - X_F(1)} P(X|C_1) dX \\ &= P_{C_2} \int_{X_F(1)} P(X|C_2) dX + P_{C_1} \int_{X_F} P(X|C_1) dX - P_{C_1} \int_{X_F(1)} P(X|C_1) dX \\ &= P_{C_1} + P_{C_2} \int_{X_F(1)} P(X|C_2) dX - P_{C_1} \int_{X_F(1)} P(X|C_1) dX \end{aligned}$$

and we can see that  $P_E$  is minimized when

$$\int_{X_F(1)} [P(X|C_2)P_{C_2} - P(X|C_1)P_{C_1}] dX$$

is minimized, which occurs by defining

$$X_F(1) = [X: P(X|C_1)P_{C_1} > P(X|C_2)P_{C_2}] \quad (A-3)$$

It follows from (A-1) and (A-3) that

$$X_F(2) = X_F - X_F(1) = [X: P(X|C_1)P_{C_1} < P(X|C_2)P_{C_2}].$$

Hence, the classification rule  $d(X)$  can be expressed, defining the discriminant

function  $\Lambda_1(X) = \frac{P(X|C_1)P_1}{P(X|C_2)P_2}$ , as follows:

$$d(X): \Lambda_1(X) \begin{matrix} C_1 \\ > \\ < \\ C_2 \end{matrix} 1$$

or taking logs, we have the equivalent form

$$d(X): \Lambda_2(X) = \ln \frac{p(X|C_1)}{p(X|C_2)} + \ln \left( \frac{P_1}{P_2} \right) \begin{matrix} > \\ < \end{matrix} \begin{matrix} C_1 \\ C_2 \end{matrix} 0. \quad (A-4)$$

Since

$$\frac{p(X|C_1)}{p(X|C_2)} = \frac{|V_{X,2}|^{1/2}}{|V_{X,1}|^{1/2}} \exp \left[ -\frac{1}{2} (X-m_1)^T V_{X,1}^{-1} (X-m_1) - (X-m_2)^T V_{X,2}^{-1} (X-m_2) \right],$$

where  $|V_{X,i}| = \det V_{X,i}$ , we have

$$\ln \left( \frac{p(X|C_1)}{p(X|C_2)} \right) = \frac{1}{2} \ln \left( \frac{|V_{X,2}|}{|V_{X,1}|} \right) + \frac{1}{2} (X-m_2)^T V_{X,2}^{-1} (X-m_2) - \frac{1}{2} (X-m_1)^T V_{X,1}^{-1} (X-m_1)$$

and since  $V_{X,i}^{-1}$  is symmetric,  $i = 1, 2$ ,

$$(X-m_i)^T V_{X,i}^{-1} (X-m_i) = X^T V_{X,i}^{-1} X + m_i^T V_{X,i}^{-1} m_i - 2X^T V_{X,i}^{-1} m_i$$

$$\text{since symmetry} \quad \Rightarrow X^T V_{X,i}^{-1} m_i = m_i^T V_{X,i}^{-1} X.$$

For example,

$$\begin{aligned} \begin{bmatrix} X_1, X_2 \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} m_{i1} \\ m_{i2} \end{bmatrix} &= \begin{bmatrix} X_1, X_2 \end{bmatrix} \begin{bmatrix} am_{i1} + bm_{i2} \\ bm_{i1} + cm_{i2} \end{bmatrix} \\ &= X_1(am_{i1} + bm_{i2}) + X_2(bm_{i1} + cm_{i2}) \end{aligned}$$

and

$$\begin{aligned} \begin{bmatrix} m_{i1}, m_{i2} \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} &= \begin{bmatrix} m_{i1}, m_{i2} \end{bmatrix} \begin{bmatrix} aX_1 + bX_2 \\ bX_1 + cX_2 \end{bmatrix} \\ &= m_{i1}(aX_1 + bX_2) + m_{i2}(bX_1 + cX_2) \\ &= X_1(am_{i1} + bm_{i2}) + X_2(bm_{i1} + cm_{i2}). \end{aligned}$$

Then,

$$\begin{aligned} \ln \left( \frac{p(X|C_1)}{p(X|C_2)} \right) &= \frac{1}{2} \ln \left[ \frac{|V_{X,2}|}{|V_{X,1}|} \right] + \frac{1}{2} X^T V_{X,2}^{-1} X + \frac{1}{2} m_2^T V_{X,2}^{-1} m_2 \\ &\quad - \frac{1}{2} \cdot 2 X^T V_{X,2}^{-1} m_2 - \frac{1}{2} X^T V_{X,1}^{-1} X - \frac{1}{2} m_1^T V_{X,1}^{-1} m_1 \\ &\quad + \frac{1}{2} \cdot 2 X^T V_{X,1}^{-1} m_1 \end{aligned}$$

and Equation (A-4) may be written as

$$\begin{aligned}
\Lambda_2(X) = & \ln \frac{P_1}{P_2} + \frac{1}{2} \ln \frac{|V_{X,2}|}{|V_{X,1}|} + \frac{1}{2} \left[ X^T (V_{X,2}^{-1} - V_{X,1}^{-1}) X \right] \\
& + X^T (V_{X,1}^{-1} m_1 - V_{X,2}^{-1} m_2) + \frac{1}{2} m_2^T V_{X,2}^{-1} m_2 \\
& - \frac{1}{2} m_1^T V_{X,1}^{-1} m_1 .
\end{aligned} \tag{A-5}$$

Note that  $\Lambda_2(X) = 0 \Rightarrow$  Equation (A-5) defines a quadratic surface

$$\begin{aligned}
& \left[ \frac{1}{2} X^T (V_{X,2}^{-1} - V_{X,1}^{-1}) X + X^T (V_{X,1}^{-1} m_1 - V_{X,2}^{-1} m_2) \right. \\
& \quad \left. + \frac{1}{2} (m_2^T V_{X,2}^{-1} m_2 - m_1^T V_{X,1}^{-1} m_1) \right] \\
& = \ln \frac{P_2}{P_1} + \frac{1}{2} \ln \frac{|V_{X,1}|}{|V_{X,2}|} \\
& = \text{constant}.
\end{aligned} \tag{A-6}$$

If  $V_X = V_{X,1} = V_{X,2}$ ,  $\Lambda_2(X)$  reduces to  $\Lambda_2^L(X)$

with

$$\Lambda_2^L(X) = \ln \frac{P_1}{P_2} + X^T (V_X^{-1}) (m_1 - m_2) + \frac{1}{2} (m_2 + m_1)^T V_X^{-1} (m_2 - m_1) \tag{A-7}$$

and  $\Lambda_2^L(X) = 0$  implies a planar (or linear) decision surface;



defining

$$W^T = (W_1, \dots, W_n)^T = V_X^{-1}(m_2 - m_1)$$

$$\ln \frac{P_1}{P_2} - X^T W + \frac{1}{2} (m_2 + m_1)^T W = 0$$

$$\text{or} \quad X^T W - W_0 = 0 \quad (\text{A-8})$$

$$\text{with} \quad W_0 = + \frac{1}{2} (m_2 + m_1)^T W + \ln \frac{P_1}{P_2}$$

and Equation (A-8) is the equation for a plane in  $X_F = R^n$

with perpendicular given by  $\frac{W_a}{\|W\|_I}$  for  $\|W\|_I = \sqrt{\sum_{i=1}^n W_i^2}$  with  $\frac{W_0}{\|W\|_I} = \text{distance}$  to plane from origin and  $W_a = (W_1, W_2, \dots, W_n, W_0)$ .

Thus, in this special case, the optimal decision surface is a linear discriminant, since  $\Lambda_2^L$  is a linear combination of feature values. That  $\Lambda_2^L(X) = 0$  defines a linear discriminant for the case  $n = 2$  is shown as follows:

$$(X) = (X_1, X_2) \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} - W_0 = 0 \Rightarrow$$

$$W_1 X_1 + W_2 X_2 - W_0 = 0$$

$$\frac{W_1}{\|W\|_I} X_1 + \frac{W_2}{\|W\|_I} X_2 - \frac{W_0}{\|W\|_I} = 0$$

with

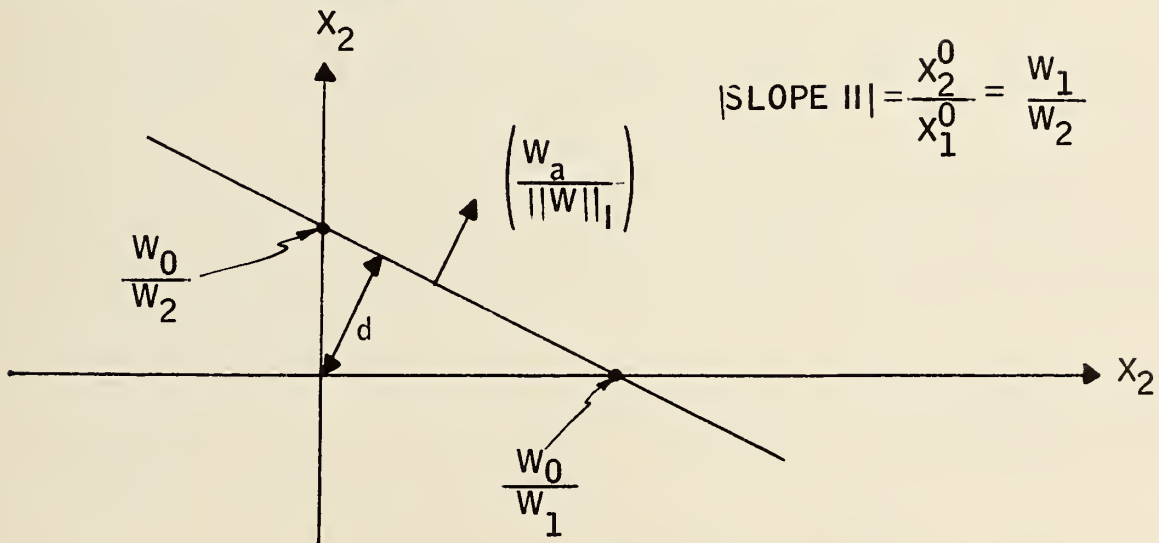
$$\|W\|_I = \sqrt{W_1^2 + W_2^2}.$$

Solving for  $X_2$ ,

$$X_2 = \left( -\frac{W_1}{||W||_I} X_1 + \frac{W_0}{||W||_I} \right) \frac{||W||_I}{W_2}$$

giving a line in two space with slope  $-\frac{W_1}{W_2}$

and  $X_2$  intercept at  $\frac{W_0}{W_2}$ ,  $X_1$  intercept at  $\frac{W_0}{W_1}$ .



Note that the distance from the plane to the origin is

$$d = \frac{W_0}{||W||_I} = \frac{W_0}{\sqrt{W_1^2 + W_2^2}}.$$

In general, a point  $X$  on the hyperplane defined by  $W_a = (W_1, W_2, \dots, W_n, W_0)$  must satisfy  $[X = (X_1, \dots, X_n)]$

$$(X_1, X_2, \dots, X_n) \cdot (W_1, W_2, \dots, W_n) = W_0.$$

If we look at a point  $X$  on the hyperplane as a vector  $\vec{X}$  from the origin to point  $X$ , the angle  $\theta$  between the vector  $\vec{W}$  and  $\vec{X}$  is given by [using  $\vec{W} = (W_1, W_2, \dots, W_n)$ ]

$$\cos \theta = \frac{\vec{X} \cdot \vec{W}}{||\vec{X}||_I ||\vec{W}||_I}$$

For  $\theta = 0^\circ$ ,  $\vec{X}$  is the perpendicular from the origin to the hyperplane and we have

$$\frac{\vec{X} \cdot \vec{W}}{||\vec{X}||_I ||\vec{W}||_I} = 1 \Rightarrow d = ||\vec{X}||_I = \frac{\vec{X} \cdot \vec{W}}{||\vec{W}||_I}.$$

But since  $X$  is assumed to be a point on the hyperplane,

$$\vec{X} \cdot \vec{W} = W_0$$

hence,

$$d = \frac{W_0}{||\vec{W}||_I}.$$

# USING EQUATION (A-7) WHEN ONLY EXPERIMENTAL DATA IS AVAILABLE

The problem with using Equation (A-7) directly is that, for two pattern classes,  $V_{x,1} = V_{x,2}$  is often not true. The problem becomes:

approximate  $\Lambda_2(x)$  in (A-5) by a linear  
discriminant  $\hat{\Lambda}_2^L(x)$  that is of the form of  
 $\Lambda_2^L(x)$  given in (A-7).

The reason we want to make this approximation is to allow simple implementation of the classifier. If we choose, using (A-7) as a guide,

$$\hat{\Lambda}_2^L(x) = X^T \hat{V}_{12}^{-1} (\hat{m}_2 - \hat{m}_1) - \frac{1}{2} (\hat{m}_2 + \hat{m}_1)^T \hat{V}_{12}^{-1} (\hat{m}_2 - \hat{m}_1) - K_L \begin{matrix} C_2 \\ > \\ C_1 \end{matrix} 0$$

where  $K_L$  is adjusted to minimize the error probability (since the actual a priori probabilities  $P_1$  and  $P_2$  are seldom known) on the experimental data comprising the SDD.<sup>+</sup> Note also that  $\hat{V}_{12}$  and  $\hat{m}_1, \hat{m}_2$  must be estimated using the experimental data.

A reasonable\* choice for  $\hat{V}_{12}$  is the average covariance over both classes  $C_1$  and  $C_2$ . This is given by

$$\hat{V}_{12} = \rho_1 \hat{V}_1 + \rho_2 \hat{V}_2 + \rho_1 \rho_2 (\hat{m}_1 - \hat{m}_2)(\hat{m}_1 - \hat{m}_2)^T$$

where  $\hat{V}_1, \hat{m}_1, \hat{V}_2, \hat{m}_2$  are computed from the experimental data using Equations 2.1-2 a and b; further, we use

$$\rho_1 = \frac{M_1}{M_1 + M_2} \quad \rho_2 = \frac{M_2}{M_1 + M_2} \quad .$$

\* See Chapter 2, pages 29 and 30, Equations (13) through (16). This Minimum Mean Square estimate of the coefficients gives a discriminant equivalent to using  $\hat{V}_{12}$  as above.

+ Sample Data Distribution

Then we have

$$\hat{\Lambda}_2^L(X) = (X - \frac{1}{2}\hat{m}_2 - \frac{1}{2}\hat{m}_1)^T \hat{V}_{12}^{-1} (\hat{m}_2 - \hat{m}_1) - K_L \begin{matrix} C_2 \\ > \\ C_1 \end{matrix} 0 \quad (A-9a)$$

One may also use

$$\hat{\Lambda}_2^L(x) = (X - \rho_2 \hat{m}_2 - \rho_1 \hat{m}_1)^T \hat{V}_{12}^{-1} (\hat{m}_2 - \hat{m}_1) - K_L \begin{matrix} C_2 \\ > \\ C_1 \end{matrix} 0 \quad (A-9b)$$

instead of (A-9a) in order to reflect the fact that one mean value has been computed using a larger  $M_i$  (larger  $\rho_i$ ) than the other. This weighted sum is then dependent on the samples taken; if  $M_1 = M_2$ , (A-9b) reduces to (A-9a).

From (A-2) (the expression for  $P_E$ ), and Equation (A-7), the expression for  $\Lambda_2^L(x)$  obtained from  $\Lambda_2(x)$  when  $V_{x,1} = V_{x,2} = V_x$ , it can be shown (see Van Trees vol. 1, pg. 97-99, pg. 36-38) that  $P_E$  is completely determined by the Mahalanobis distance

$$d_m^2(m_1, m_2) = (m_1 - m_2)^T V_x^{-1} (m_1 - m_2) .$$

Further, it is shown that as  $d_m^2$  increases,  $P_{E2}$  decreases. Hence, inter-class complexity is inversely proportional to  $d_m^2$ . A reasonable measure of complexity with respect to the decision rule given by (A-9b) or (A-9a) is then

$$\hat{d}_m(\hat{m}_1, \hat{m}_2) = (\hat{m}_1 - \hat{m}_2)^T \hat{V}_{12}^{-1} (\hat{m}_1 - \hat{m}_2)$$

which is inversely proportional to inter-class complexity.



If the features are statistically independent,

$$\hat{V}_{12} = \begin{bmatrix} \hat{\sigma}_1^2 & 0 & \dots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \hat{\sigma}_n^2 \end{bmatrix}$$

we have

$$\hat{d}_m^2(\hat{m}_1, \hat{m}_2) = \sum_{i=1}^n \frac{\Delta \hat{m}_i^2}{\hat{\sigma}_i^2} \quad (\text{A-10})$$

Thus, for ease in computation, the measure of complexity is taken as (A-10), using  $\Delta \hat{m}_i^2 = (\hat{m}_{1i} - \hat{m}_{2i})^2$ , with

$$\hat{\sigma}_i^2 = \rho_1 \hat{\sigma}_{1i}^2 + \rho_2 \hat{\sigma}_{2i}^2$$

from the form of  $\hat{V}_{12}$ .

## APPENDIX B

### PIECEWISE LINEAR DISCRIMINANTS

Assume the multimodal Gaussian classes to be distinguished have densities given by

Class 1:

$$p(X|C_1) = \sum_{j=1}^{K_1} \frac{P_1^j}{P_1} p(X|C_1^j)$$

Class 2:

$$p(X|C_2) = \sum_{j=1}^{K_2} \frac{P_2^j}{P_2} p(X|C_2^j)$$

with

$$\sum_{j=1}^{K_i} P_i^j = P_i, \text{ and } X \text{ an } n\text{-vector.}$$

As in Appendix A, we define a minimum probability of error classification rule, that is, a choice of  $X_F(1)$  and  $X_F(2)$  such that

$$P_E = P_1 \int_{X_F(2)} p(x|C_1) dX + P_2 \int_{X_F(1)} p(X|C_2) dX$$

is minimized. Using the same reasoning as in Appendix A, the minimum  $P_E$  decision rule is

$$d(X): \begin{matrix} C_2 \\ < \\ \Lambda(X) > \\ C_1 \end{matrix} 1.$$

where

$$\Lambda(X) = \frac{P_1 p(X|C_1)}{P_2 p(X|C_2)} \begin{matrix} < \\ > \end{matrix} \begin{matrix} C_2 \\ C_1 \end{matrix} 1. \quad (\text{B-1a})$$

However,

$$\Lambda(X) = \frac{P_1 \sum_j \frac{P_1^j}{P_1} p(X|C_1^j)}{P_2 \sum_j \frac{P_2^j}{P_2} p(X|C_2^j)} \quad (\text{B-1b})$$

which cannot be reduced to a simple expression as was done in Appendix A.

To get the desired simple form for the decision rules, a suboptimal procedure is used. Extend the original 2-class problem to a  $K_1 + K_2$  class problem by considering each mode of the mixture densities as a separate class. In this case, we desire the minimum probability of error for the  $K_1 + K_2$  class problem,

$$P_E = \sum_{i=1}^2 P_i \sum_{j=1}^{K_i} \frac{P_i^j}{P_i} \sum_{\substack{K=1 \\ K \neq j+(i-1)K_i}}^{K_1+K_2} \int_{X_F(K)} p(X|C_i^j) dX \quad (\text{B-2})$$

$$= \sum_{j=1}^{K_1+K_2} P_j \sum_{\substack{K=1 \\ K \neq j}}^{K_1+K_2} \int_{X_F(K)} p(X|\hat{C}_j) dX$$

where

$$\hat{C}_j = \begin{cases} C_1^j; & j \leq K_1 \\ C_2^{j-K_1}; & j \geq K_1+1 \end{cases}$$

$$P_j = \begin{cases} P_1^j; & j \leq K_1 \\ P_2^{j-K_1}; & j \geq K_1+1. \end{cases}$$

Since the  $X_F(K)$  are assumed disjoint and  $\bigcup_K X_F(K) = X_F$ , we have

$$\sum_{\substack{K=1 \\ K \neq 2}}^{K_1 + K_2} \int_{X_F(K)} p(X | \hat{C}_j) dx = \int_{X_F - X_F(j)} p(X | \hat{C}_j) dx \quad (B-3)$$

where the decision rule

$$\hat{d}(x) = \hat{C}_j \quad \text{For } X \in X_F(j)$$

is to be minimized with respect to  $P_E$ .

We have

$$\begin{aligned}
 P_E &= + \sum_{j=1}^{K_1+K_2} P_j \int_{X_F - X_F(j)} p(X | \hat{C}_j) dX \\
 &= \sum_{j=1}^{K_1+K_2} P_j \int_{X_F} p(X | \hat{C}_j) dX - \sum_{j=1}^{K_1+K_2} P_j \int_{X_F(j)} p(X | \hat{C}_j) dX \\
 &= 1 - \sum_{j=1}^{K_1+K_2} P_j \int_{X_F(j)} p(X | \hat{C}_j) dX
 \end{aligned} \tag{B-4}$$

and to minimize  $P_E$ , maximize

$$\epsilon = \sum_{j=1}^{K_1+K_2} P_j \int_{X_F(j)} p(X | \hat{C}_j) dX.$$

To maximize  $\epsilon$ , choose

$$X_F(j) = \{ X \in X_F : P_j p(X | \hat{C}_j) \geq P_K p(X | \hat{C}_K), K=1, \dots, K_1+K_2 \}.$$

This gives the decision rule  $\hat{d}(X)$  in the form

$$\hat{d}(X): \text{ decide } \hat{C}_j \text{ for } j \ni P_j p(X | \hat{C}_j) = \max_K \{ P_K p(X | \hat{C}_K), K=1, \dots, K_1+K_2 \}.$$

Since the maximum can be determined by a pairwise comparison of the

$$\{ P_j p(X | \hat{C}_j) \},$$

the decision rule  $\hat{d}(X)$  in algorithmic form is:



ALGORITHM 1 ( $\hat{d}(X)$ )

[0]  $\alpha \leftarrow P_1 p(X | \hat{C}_1)$

$J \leftarrow 1$

$I \leftarrow 1$

[1]  $I \leftarrow I + 1$

if  $I > K_1 + K_2$ , go to [4]

[2] if  $\frac{\alpha}{P_I p(X | \hat{C}_I)} > 1$  go to [1]

otherwise go to [3]

[3]  $J \leftarrow I$

$\alpha \leftarrow P_J p(X | \hat{C}_J)$

go to [1]

[4] decide class  $\hat{C}_J \Rightarrow$  decide class  $C_1^J$  if  $J \leq K_1$ ,  $C_2^{J-K_1}$  otherwise.  
Stop.

Note that each test in step [2] above is of the form

$$\Lambda_1(X) = \frac{P_J p(X | \hat{C}_J)}{P_I p(X | \hat{C}_I)} \begin{matrix} > 1 \\ < 1 \end{matrix} \begin{matrix} \{\hat{C}_1, \dots, \hat{C}_{K_1+K_2}\} - \{\hat{C}_I\} \\ \{\hat{C}_1, \dots, \hat{C}_{K_1+K_2}\} - \{\hat{C}_J\} \end{matrix} \quad (B-5)$$

which gives the same results as in the Appendix A expressions for  $\Lambda_2(X)$ ,

and can be approximated by  $\hat{\Lambda}_2^L(X)$ . When  $\hat{\Lambda}_2^L(X)$  is used the result is a

linear discriminant for each pair of the  $(K_1 + K_2)$  subclasses. The resulting decision surface for the original two class problem is then piecewise linear. Note that we are not interested in distinguishing between the subclasses of class 1, or of class 2, only between the subclass of 1 and the subclass of 2. Thus the algorithm can be modified as follows, resulting in fewer tests as follows: (Algorithm 1 requires  $K_1 + K_2 - 1$  tests)

ALGORITHM 2 ( $\hat{d}(X)$ )

- [ 0]  $\alpha \leftarrow P_1^{I_1} p(X|C_1^{I_1})$   
 $JC = 1$   
 $I_2 \leftarrow 0$   
 $I_1 \leftarrow 1$
- [ 1]  $I_2 \leftarrow I_2 + 1$   
 Go to [ 3] unless  $I_2 > K_2$ ; then go to [ 6]
- [ 2]  $I_1 \leftarrow I_1 + 1$   
 Go to [ 4] unless  $I_1 > K_1$ ; then go to [ 6]
- [ 3]  $\beta \leftarrow P_2^{I_2} p(X|C_2^{I_2})$ ; go to [ 5]
- [ 4]  $\alpha \leftarrow P_1^{I_1} p(X|C_1^{I_1})$ ; go to [ 5]
- [ 5] If  $\frac{\alpha}{\beta} > 1$   $JC = 1$ , go to [ 1]  
 otherwise  $JC = 2$ , go to [ 2]
- [ 6] Decide  $C_{JC}$   
 [stop]

The decision rule  $\hat{d}(X)$  given by Algorithm 1 and B-5, while optimal (minimum PE) for the  $K_1 + K_2$  class problem, is suboptimal for the two class problem of interest (thus Alg. 2 is suboptimal). This follows since deciding  $C_2$  based on Algorithm 1 ( $C_J$ ,  $J > K_1$ ) ensures only that

$$P_J p(X | C_J) = \max_I \{ P_I p(X | C_I) \};$$

it is still possible for

$$\sum_{j=1}^{K_1} P_1^j p(X | C_1^j) > \sum_{j=1}^{K_2} P_2^j p(X | C_2^j),$$

violating B-1a.

## APPENDIX C

### CTC PROGRAM COMPUTATIONS

#### STATISTICAL SCATTER COMPUTATIONS (using notation of Chapter 3)

Means ( $X_i(j)$  is an n-vector  $\Rightarrow m_i$  an n-vector)

- 1) class mean

$$m_i = \frac{1}{M_i} \sum_{j=1}^{M_i} X_i(j) \quad (i = 1, \dots, N)$$

- 2) total mean

$$m_{TOT} = \sum_{i=1}^n P_i m_i$$

#### Dispersions (matrices)

- 1) within class

$$\delta_i = \frac{1}{M_i} \sum_{j=1}^{M_i} X_i(j) X_i^T(j)$$

- 2) mean between class

$$\delta_B = \sum_{i=1}^N P_i m_i m_i^T$$

3) mean within class

$$\delta_w = \sum_{i=1}^N P_i \delta_i$$

### Covariances (matrices)

1) within class

$$V_i = \delta_i - m_i m_i^T$$

2) mean between class

$$V_B = \delta_B - m_{TOT} m_{TOT}^T$$

3) average within class

$$\bar{V} = \sum_{i=1}^N P_i V_i$$

4) mean within class

$$V_W = \delta_W - m_{TOT} m_{TOT}^T$$

5) total

$$V_T = V_B + \bar{V}$$



### Feature Standard Deviations (j, i = 1, . . . , n) - scalars

- 1) within class

$$\sigma_i(j) = \sqrt{V_i(j, j)}$$

- 2) mean between class

$$\sigma_B(j) = \sqrt{V_B(j, j)}$$

- 3) average within class

$$\bar{\sigma}(j) = \sqrt{\bar{V}(j, j)}$$

- 4) total

$$\sigma_T(j) = \sqrt{V_T(j, j)}$$

### Feature Correlations (scalar)

- 1) within class

$$\rho_i(j, k) = \frac{V_i(j, k)}{\sigma_i(j) \sigma_i(k)}$$

- 2) mean between class

$$\rho_B(j, k) = \frac{V_B(j, k)}{\sigma_B(j) \sigma_B(k)}$$

3) average within class

$$\bar{\rho}(j, k) = \frac{\bar{V}(j, k)}{\bar{\sigma}(j) \bar{\sigma}(k)}$$

4) total

$$\rho_T(j, k) = \frac{V_T(j, k)}{\sigma_T(j) \sigma_T(k)}$$

Reduction of Formulae

$$\bar{V} = \sum P_i V_i = \sum P_i (\delta_i - m_i m_i^T) = \sum P_i \delta_i - \sum P_i m_i m_i^T$$

$$V_W = \sum P_i \delta_i - (\sum P_i m_i) (\sum P_i m_i)^T$$

$$V_B = \sum P_i m_i m_i^T - (\sum P_i m_i) (\sum P_i m_i)^T$$

$$\begin{aligned} V_T &= V_B + \bar{V} = \sum P_i \delta_i - \sum P_i m_i m_i^T + \sum P_i m_i m_i^T - (\sum P_i m_i) (\sum P_i m_i)^T \\ &= V_W \end{aligned}$$

$V_W$  serves as a double check on accuracy in program computations.

DISTANCE COMPUTATION IN CTC

$$1) \hat{m}(i) = \frac{1}{M_i} \sum_{j=1}^{M_i} X_i(j)$$

$$2) \quad m_{TOT} = \sum_{j=1}^N P_i \hat{m}(i)$$

$$3) \quad m_r(i) = \hat{m}(i) - m_{TOT}$$

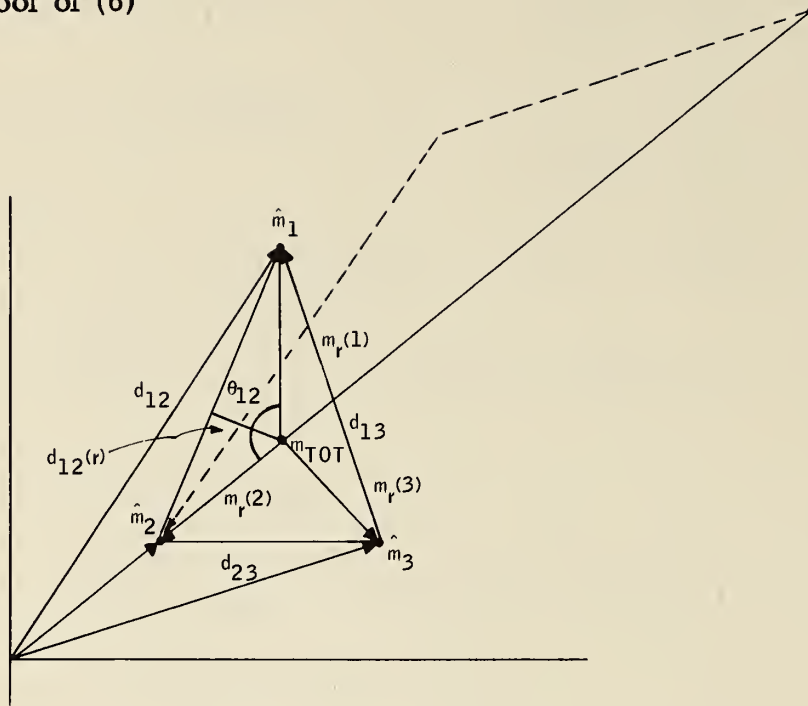
$$4) \quad \alpha(i, j) = m_r^T(i) I m_r(j) ; \quad ||m_r(i)||_I^2 \triangleq \alpha(i, i)$$

$$5) \quad \beta(i, j) = \alpha(i, j) / \sqrt{\alpha(i, i)} \sqrt{\alpha(j, j)} = \cos \theta_{ij} \text{ since } \cos \theta = \frac{\vec{X} \cdot \vec{Y}}{||\vec{X}||_I ||\vec{Y}||_I}$$

$$\begin{aligned} 6) \quad \tilde{d}_E^2(i, j) &= \alpha(i, i) + \alpha(j, j) - 2 \sqrt{\alpha(i, i)} \sqrt{\alpha(j, j)} \beta(i, j) \\ &= ||m_r(i)||_I^2 + ||m_r(j)||_I^2 - 2 m_r^T(i) I m_r(j) \\ &= ||m_r(i) - m_r(j)||_I^2 \\ &= ||\hat{m}(i) - \hat{m}(j)||_I^2 \end{aligned}$$

$$7) \quad \theta(i, j) = \text{ATAN} \left[ \frac{\sqrt{1 - \beta^2(i, j)}}{\beta(i, j)} \right] * \frac{180}{\pi}$$

(8) Proof of (6)



$$d_{12}^2 = \|\hat{m}_1 - \hat{m}_2\|_I^2 = \|\hat{m}_1\|_I^2 + \|\hat{m}_2\|_I^2 - 2 \hat{m}_1 \cdot \hat{m}_2$$

$$\therefore d_E^2(i, j) = \|\hat{m}_i\|_I^2 + \|\hat{m}_j\|_I^2 - 2 \hat{m}_i \cdot \hat{m}_j$$

$$= \hat{m}_i^T I \hat{m}_i + \hat{m}_j^T I \hat{m}_j - 2 \hat{m}_i^T I \hat{m}_j$$

$$\tilde{d}_E^2(i, j) = \|m_r(i) - m_r(j)\|^2 = m_r^T(i) I m_r(i) + m_r^T(j) I m_r(j)$$

$$- 2 m_r^T(i) I m_r(j)$$

$$\text{and } \hat{m}_r(\alpha) = \hat{m}_\alpha - \hat{m}_{TOT}$$

$$\begin{aligned}
\tilde{d}^2(i, j) &= (\hat{m}_i - m_{TOT})^T I(\hat{m}_i - m_{TOT}) + (\hat{m}_j - m_{TOT})^T I(\hat{m}_j - m_{TOT}) \\
&\quad - 2 (\hat{m}_j - m_{TOT})^T I(\hat{m}_j - m_{TOT}) \\
&= \hat{m}_i^T I \hat{m}_i + m_{TOT}^T I m_{TOT} - 2 m_{TOT}^T I \hat{m}_i \\
&\quad + \hat{m}_j^T I \hat{m}_j + m_{TOT}^T I m_{TOT} - 2 m_{TOT}^T I \hat{m}_j \\
&\quad - 2 \hat{m}_i^T I \hat{m}_j - 2 m_{TOT}^T I m_{TOT} + 2 m_{TOT}^T I \hat{m}_j \\
&\quad + 2 \hat{m}_i^T I m_{TOT} \\
&= \|\hat{m}_i\|_I^2 + \|\hat{m}_j\|_I^2 + 2 \|m_{TOT}\|_I^2 - 2 \|m_{TOT}\|_I^2 \\
&\quad - 2 m_{TOT}^T I (\hat{m}_i + \hat{m}_j) + 2 m_{TOT}^T I (\hat{m}_j + \hat{m}_i) \\
&\quad - 2 \hat{m}_i^T I \hat{m}_j \\
&= \|\hat{m}_i\|_I^2 + \|\hat{m}_j\|_I^2 - 2 \hat{m}_i^T I \hat{m}_j = d_E^2(i, j)
\end{aligned}$$

## PROJECTIONS ONTO SUBSPACES OF $X_F$

Program computes projections as follows: given a transformation matrix (m rows x n columns) A, and an n-vector  $X \in X_F$ , the transformation is (onto the subspace defined by A)

$$Y = A^T (AA^T)^{-1} A X$$



where  $Y$  is an  $n$ -vector. Projections onto the complement of the subspace defined by  $A$  are computed using

$$Y = X - A^T (AA^T)^{-1} A X.$$

Note that the program allows scaling of the results, i. e. ,

$$Y_S = Y \cdot \text{scale factor}.$$

APPENDIX D  
LISTING OF THE CTC PROGRAM

```

1: C *****
2: C      CASCADED THRESHOLD CLASSIFIER PROGRAM
3: C *****
4: C
5: C      CARD INPUT
6: C      LINEPRINTER OUTPUT
7: C      TYPEWRITER MONITOR
8: C
9: C *****
10: C
11: C      SUBROUTINES          N IS MATRIX OR VECTOR SIZE
12: C
13: C      DMDPRD(A,B,C,D,N)    MATRIX PRODUCT D=A*B*C
14: C      MPRD(A,B,C,N(L))    MATRIX PRODUCT C=A*B
15: C                          A IS N(1) BY N(2), B IS N(2) BY N(3),
16: C      MSUM(S1,A,S2,B,C,N)  MATRIX SUM C=S1*A + S2*B
17: C                          S1, S2 SCALARS
18: C      MVPRD(A,B,C,N)       MATRIX VECTOR PRODUCT C=A*B
19: C                          A IS N BY N, B IS N BY 1
20: C      SEVAL(A,EV,LAMBDA,B,IV,N) SMALLEST EIGENVALUE LAMBDA AND
21: C                          CORRESPONDING EIGENVECTOR EV OF A
22: C      TDINVR(K1,K2,NR,NC,A,MNR,IWA,DET)
23: C                          MATRIX A NR BY NC IS RETURNED INVERTED
24: C                          AND DETERMINANT DET RETURNED
25: C                          DETERMINANT DET RETURNED
26: C      TRADEOFF(NC,NF,NFV,FEAT,TV1,B,TV2,NS,SCORE,M0DE)
27: C                          TABULAR LISTING OF TRADEOFF CURVE DATA
28: C                          ORDERED BY SCORE
29: C      TRANSP(A,B,N)        B=A TRANSPOSE
30: C      VIPRD(A,B,C,N)       VECTOR INNER PRODUCT C=A*B
31: C                          C IS SCALAR
32: C      VOPRD(A,B,C,N)       VECTOR OUTER PRODUCT C=A*B
33: C                          C IS N BY N MATRIX
34: C      VSUM(S1,A,S2,B,C,N)  VECTOR SUM C=S1*A + S2*B
35: C
36: C
37: C      ARRAYS
38: C
39: C      BCOV--BETWEEN CLASS COVARIANCE (COVARIANCE OF THE CLASS MEANS)
40: C      CLAS--TYPE TO CLASS ASSIGNMENT
41: C      COV--CLASS COVARIANCES
42: C      DISP--CLASS DISPERSIONS
43: C      FEAT--FEATURE VECTORS AND IDENTIFICATION FIELDS
44: C      FLAG--PROGRAM CONTROL FLAGS
45: C      INCARD--CARD INPUT BUFFER
46: C      ITV--NUMBER AND SCALE FACTOR FOR NEW FEATURE DEFINITIONS
47: C      ITVP--TEMPORARY VECTOR
48: C      LIST--LINE PRINTER LIST OPTIONS
49: C      NS--NUMBER OF SAMPLES IN CLASSES
50: C      PR0B--WEIGHT FACTORS FOR CLASSES
51: C      SAVE--INCORE STORAGE AREA
52: C      SCORE--TRADEOFF SCORES, ALSO USED FOR BOOLEAN BIT STRINGS
53: C      SFEAT--SELECTED FEATURES
54: C      TCOV--TOTAL COVARIANCE
55: C      TM1--TEMPORARY MATRIX, ALSO USED FOR COEFFICIENTS FOR NEW FEATURE
56: C          DEFINITIONS
57: C      TM2--TEMPORARY MATRIX
58: C      TV1--TEMPORARY VECTOR, ALSO USED AS CONSTANT FOR NEW FEATURE DEFINITIONS
59: C      TV2--TEMPORARY VECTOR

```

```

60: C      TV3--TEMPORARY VECTOR
61: C      WCBV--WITHIN CLASS COVARIANCE (AVERAGE CLASS COVARIANCE)
62: C      XBAR--CLASS MEANS
63: C      XBART--TOTAL MEAN
64: C
65: C      PARAMETERS
66: C
67: C      DATE--PROGRAM DATE
68: C      ID--PROGRAM IDENTIFIER--FIRST FOUR CHARACTERS
69: C      ID1--PROGRAM IDENTIFIER--LAST FOUR CHARACTERS
70: C      KMODE--CLASSIFIER METHOD
71: C      MODE2--SECONDARY FEATURE TRANSFORMATION FLAG
72: C      NC--NUMBER OF CLASSES, MAXIMUM OF 10
73: C      NCT--NUMBER OF CONTROL CARD INPUT CODES
74: C      NDT--NUMBER OF DATA CARD INPUT CODES
75: C      NF--NUMBER OF FEATURES, MAXIMUM OF 10
76: C      NFV--NUMBER OF FEATURE VECTORS, MAXIMUM OF 200
77: C      NPCN--PROJECTION FLAG, #1 FOR SUBSPACE, #2 FOR COMPLEMENT
78: C      NSAV--20*NR. OF VECTORS SAVED
79: C      NSS--NUMBER OF SAMPLES SELECTED
80: C      NT--NUMBER OF TYPES, MAXIMUM OF 20
81: C
82: C*****
83: C
84: C      COMMON SAVE(4000),NSAV
85: C      COMMON INCARD(19),FEAT(20,200),PRBB(10),CLAS(20),SFEAT(10)
86: C      1  ,NS(0:10),XBAR(10,10),TM1(10,10),TM2(10,10),DISP(10,10,10)
87: C      2  ,CBV(10,10,10),TCBV(10,10),WCBV(10,10),BCBV(10,10),XBART(10)
88: C      3  ,TV1(10),TV2(10),TV3(10),ITV2(10),SCORE(200)
89: C      INTEGER LIST(20),FLAG(20),ITV(20)
90: C      EQUIVALENCE(ITV,TV2)
91: C      INTEGER FEAT,CLAS,SFEAT,DATE,SAVE,FLAG
92: C
93: C      PRESTORE PARAMETERS
94: C      NAME LIST
95: C
96: C      REMOVE NEXT STATEMENT FOR NORMAL PROGRAM EXECUTION
97: C      CALL RESTART
98: C
99: C      INITIALIZE
100: C
101: C      NDT=5
102: C      NCT=8
103: C      NFV=0
104: C      NC=0
105: C      NT=0
106: C      NF=0
107: C      KMODE=0
108: C      NPCN=0
109: C      DATE=0
110: C      ID=0
111: C      ID1=0
112: C      MODE2=0
113: C      NSS=0
114: C      DO 12 I=1,20
115: C 12  LIST(I),FLAG(I)=0
116: C
117: C      SETS FEAT,PRBB,CLAS,SFEAT,NS,AND XBAR((1,1) TO (4,9)) = ZERO
118: C      DO 10 I=1,4100
119: C 10  FEAT(I,1)=0
120: C

```

```

121: C*****
122: C    CONTROL LOOP
123: C*****
124: C
125: 100    GO TO 300
126: 101    CONTINUE
127: C
128: C    SECONDARY FEATURES FOR BUS DETECTOR
129: C    MODE2 = 1 OR 2 FOR TRANSFORMING FEATURES TO SECONDARY FEATURES
130: C    IF(MODE2.GT.0) GO TO 550
131: C
132: C    PROCESS COEFFICIENT CARDS
133: C    FLAG(3) = NR. OF COEFFICIENT CARDS
134: C    IF(FLAG(3).GT.0) GO TO 1100
135: C
136: C    DO DATA PREPARATION
137: C    GO TO 500
138: C
139: C    TRADEOFF LISTING OPTIONS
140: C    FLAG(4) = NR. OF FEATURE TRADEOFF LISTINGS
141: 121    IF(FLAG(4).GT.0) GO TO 1200
142: C
143: C    CALCULATE STATISTICS
144: 140    GO TO 600
145: 141    CONTINUE
146: C
147: C    CLASSIFIER OPTIONS
148: C    CLASSIFIER METHODS 1 AND 2
149: C    IF(1.LE.KMODE.LE.2) GO TO 800
150: C
151: C    CLASSIFIER METHOD 3
152: C    IF(KMODE.EQ.3) GO TO 840
153: C    GO TO 160
154: 151    CONTINUE
155: C
156: C    CLASS PAIR TRADEOFF LISTINGS
157: C    DO 155 IP=1,NC-1
158: C    DO 155 JP=IP+1,NC
159: C    IF(KMODE.LE.0) GO TO 155
160: C    WRITE(2,2307) DATE,ID,ID1
161: C    WRITE(2,2101) IP,JP
162: C    GO TO (152,153) KMODE
163: 152    CALL TRADEOFF(NC,NF,NFV,FEAT,CBV(1,IP,JP),CBV(IP,JP,JP),TV2,NS
164: C    1 ,SCORE,1)
165: C    GO TO 155
166: 153    CALL TRADEOFF(NC,NF,NFV,FEAT,CBV(1,JP,IP),CBV(JP,IP,IP),TV2,NS
167: C    1 ,SCORE,1)
168: 155    CONTINUE
169: C
170: C    ALLOW USER TO READY CARD DECK FOR NEXT INPUT
171: 160    PAUSE 1
172: C
173: C    GO TO TOP OF CONTROL LOOP
174: C    GO TO 100
175: C
176: C*****
177: C    INPUT MODULE
178: C*****
179: C
180: 300    OUTPUT(2)'INPUT',' '
181:        OUTPUT(3)'INPUT'

```



```

182: C
183: C READ CARD TO BUFFER
184: 302 READ(1,2300) I, INCARD
185: C
186: C OUTPUT CARD IMAGE EXCEPT FOR FEATURE VECTOR DATA, ID, EXECUTE
187: IF((I.NE. 1) .AND. (I.NE. 11) .AND. (I.NE. 12) .AND.
188: + (I.NE. -1) .AND. (I.NE. -99)) WRITE(2, 2316) I , INCARD
189: C
190: C CHECK FOR DATA CARD
191: C IF CODE IS 1 2 3 4 5
192: C CARD FEAT VEC CREF VEC NC NT NF
193: C SWITCH TO 310 320 330 340 350
194: C
195: IF(0.LI.I.LE.NDT) GO TO (310,320,330,340,350) I
196: C
197: C CHECK FOR FIRST FEATURE CARD FOR BUS DETECTOR
198: IF(I.EQ.11) GO TO 3100
199: C
200: C CHECK FOR SECOND FEATURE CARD FOR BUS DETECTOR
201: IF(I.EQ.12) GO TO 3110
202: C
203: C CHECK FOR CONTROL CARD
204: C IF CODE IS -1 -2 -3 -4 -5 -6 -7 -8
205: C CARD ID KMODE LIST RESTORE NFV=0 MDE2 TRADEOF NPCBN
206: C SWITCH TO 360 370 380 390 450 460 470 480
207: C
208: IF(=NCI.LE.I.LT.0) GO TO(360,370,380,390,450,460,470,480) -I
209: C
210: C CHECK FOR EXECUTE
211: IF(I.EQ.-99) GO TO 400
212: OUTPUT(2) 'CARD NOT USED'
213: C
214: C READ NEXT CARD
215: GO TO 302
216: C
217: C FEATURE VECTOR INPUT
218: C
219: 310 NFV=NFV+1
220: FEAT(1,NFV)=NFV
221: C
222: C FLAG(1) COUNTS THE NUMBER OF CARDS READ SINCE LAST EXECUTE
223: FLAG(1)=FLAG(1)+1
224: C
225: C FILL NEXT COLUMN OF FEAT ARRAY FROM CARD.
226: C FEAT(11,20,NFV) HOLDS FEATURE VECTOR
227: DEC8DE(76,2301,INCARD)(FEAT(J,NFV),J=2,7),(FEAT(J,NFV),J=11,20)
228: GO TO 302
229: C
230: C FIRST FEATURE CARD FOR BUS DETECTOR
231: C
232: 3100 NFV=NFV+1
233: FEAT(1,NFV)=NFV
234: FLAG(1)=FLAG(1)+1
235: DEC8DE(76,2301,INCARD)(FEAT(J,NFV),J=2,3),(FEAT(J,NFV),J=5,7)
236: FEAT(5,NFV)=FEAT(5,NFV)+10*FEAT(7,NFV)
237: FEAT(7,NFV)=0
238: GO TO 302
239: C
240: C SECOND FEATURE CARD FOR BUS DETECTOR
241: C
242: 3110 DEC8DE(76,2301,INCARD)(FEAT(J,NFV+1),J=11,19)

```

```

243: C
244: C SEARCH FOR MATCHING ID FROM PREVIOUSLY PROCESSED 1ST FEATURE CARD
245: D0 3112 J=NfV,1,-1
246: IF (FEAT(12,NfV+1).EQ.FEAT(2,J)) GO TO 3114
247: 3112 CONTINUE
248: OUTPUT(3) 'NO MATCH'
249: OUTPUT(2) 'NO MATCH',FEAT(11,NfV+1),FEAT(12,NfV+1)
250: GO TO 302
251: 3114 L=NfV+1
252: C
253: C CHECK FOR DUPLICATE RUNS USING A SINGLE FIRST FEATURE CARD
254: IF (FEAT(7,J).NE.0) GO TO 3120
255: C
256: C FEAT(7,J) = 0 IF NO MATCH UNTIL NOW
257: 3115 FEAT(7,J)=FEAT(11,L)
258: FEAT(4,J)=7
259: D0 3116 K=1,7
260: 3116 FEAT(10+K,J)=FEAT(12+K,L)
261: GO TO 302
262: C
263: C BUILD NEW FIRST CARD FOR DUPLICATE RUN NUMBERS
264: 3120 D0 3123 I=2,6
265: 3123 FEAT(I,NfV+1) = FEAT(I,J)
266: NfV=J=NfV+1
267: FEAT(1,J)=J
268: GO TO 3115
269: C
270: C COEFFICIENT VECTOR INPUT
271: C
272: C FLAG(3) COUNTS COEFFICIENT CARDS READ THIS TIME
273: C KT VALUES STORED IN ITV(1-10)
274: C KP VALUES STORED IN ITV(11-20)
275: C CONSTANTS STORED IN TV1(1-10)
276: C COEFFICIENTS STORED IN TM1(J,1-10)
277: C ONLY TEN COEFFICIENT CARDS CAN BE PROCESSED
278: C
279: 320 IF (FLAG(3).GE.10) GO TO 302
280: IF (Nf.EQ.0) OUTPUT(3) 'Nf=0 FOR 2 CARD'
281: I=FLAG(3)=FLAG(3)+1
282: DECODE(76,2315,INCARD) ITV(I),ITV(10+I),TV1(I),(TM1(J,I),J=1,Nf)
283: GO TO 302
284: C
285: C WEIGHT VECTOR INPUT , NORMALIZE
286: C
287: 330 DECODE(76,2302,INCARD) NC
288: C
289: C MAXIMUM NR. OF CLASSES THAT CAN BE PROCESSED IS 10
290: IF (NC.GT.10) NC=10
291: DECODE(76,2303,INCARD) J,(PRPB(J),J=1,NC)
292: C
293: C CALCULATE CLASS A-PRIORI PROBABILITIES FROM WEIGHTS ON CLASS CARD
294: A=0
295: D0 331 I=1,NC
296: 331 A=A+PRPB(I)
297: D0 332 I=1,NC
298: 332 PRPB(I)=PRPB(I)/A
299: GO TO 302
300: C
301: C CLASS ASSIGNMENT VECTOR
302: C
303: 340 DECODE(76,2302,INCARD) NT

```

```

304: C
305: C   MAXIMUM NR. OF TYPES THAT CAN BE PROCESSED IS 20
306:   IF (NT.GT.20) NT=20
307: C
308: C   CLAS(I) = CLASS NUMBER TO WHICH TYPE I IS ASSIGNED
309:   DEC9DE(76,2304,INCARD) I, (CLAS(I), I=1,NT)
310:   GO TO 302
311: C
312: C   FEATURE SELECTION VECTOR
313: C
314: 350 DEC9DE(76,2302,INCARD) NF
315: C
316: C   MAXIMUM NR. OF FEATURES THAT CAN BE USED IS 10
317:   IF (NF.GT.10) NF=10
318:   DEC9DE(76,2304,INCARD) I, (SFEAT(I), I=1,NF)
319:   GO TO 302
320: C
321: C   DATE AND ID
322: C
323: 360 DEC9DE(76,2306,INCARD) DATE, ID, ID1
324:   WRITE(3,2307) DATE, ID, ID1
325:   WRITE(2,2307) DATE, ID, ID1
326:   GO TO 302
327: C
328: C   CLASSIFIER MODE
329: C
330: 370 DEC9DE(76,2301,INCARD) KMODE, LCRUNT
331:   OUTPUT(2) LCRUNT
332:   GO TO 302
333: C
334: C   LISTING OPTIONS
335: C
336: 380 DEC9DE(76,2305,INCARD) LIST(I), I=11,20
337:   DO 388 I=11,20
338:   J=LIST(I)
339: C
340: C   SET (IF J.GT.0) OR CLEAR (IF J.LT.0) LISTING OPTION I
341:   IF (J) 382,388,384
342: 382 LIST(-J)=0
343:   GO TO 388
344: 384 LIST(J)=1
345: 388 CONTINUE
346:   GO TO 302
347: C
348: C   RESTORE SAVED DATA BEHIND NEW DATA
349: C
350: C   CHECK FOR NO FEATURE VECTORS SAVED TO RESTORE
351: 390 IF (NSAV.EQ.0) GO TO 302
352:   OUTPUT(2) 'DATA RESTORED'
353:   IF (FLAG(1).NE.0) GO TO 394
354: C
355: C   RESTORE AT BEGINNING OF FEAT ARRAY IF NO CARDS READ THIS TIME
356:   DO 392 I=1,NSAV
357: 392 FEAT(I,1)=SAVE(I)
358:   NFV=NSAV/20
359:   GO TO 302
360: C
361: C   RESTORE FOLLOWING FEATURE VECTORS LOADED BY CARDS THIS TIME
362: 394 DO 396 I=1,NSAV
363: 396 FEAT(20*NFV+I,1)=SAVE(I)
364:   NFV=NFV+NSAV/20

```

```

365:      GO TO 302
366: C
367: C      EXECUTE CARD
368: C
369: C      FLAG1 = NR. OF FEATURE VECTOR CARDS READ SINCE LAST -99
370: 400  OUTPUT(2)'EXECUTE',FLAG(1)
371: C
372: C      CHECK FOR NR. OF CLASSES, NR. OF TYPES OR NR. OF FEATURES = 0
373:      IF(NC*NT*NF.GT.0) GO TO 401
374:      OUTPUT(3) 'ERROR',NC,NT,NF
375: C
376: C      ALLOW USER TO SET SENSE SWITCH 1
377:      PAUSE 2
378: 401  CONTINUE
379: C
380: C      USER MAY INPUT PARAMS ON TYPEWRITER IF SS1 IS ON
381:      IF(SENSE SWITCH 1) 402,404
382: C
383: C      TYPEWRITER INPUT
384: C      UPPER CASE G TO TERMINATE INPUT
385: 402  INPUT(3)
386: C
387: C      DATA SUMMARY
388: C      SHORT LIST
389: 404  OUTPUT(2) NFV,NC,NT,NF,KMODE
390:      OUTPUT(2) 'PR08'
391:      WRITE(2,2309) PR08(1),I=1,NC
392:      WRITE(2,2310) I,I=1,NT
393:      WRITE(2,2311) CLAS(I),I=1,NT
394:      OUTPUT(2)'SELECTED FEATURES'
395:      WRITE(2,2312) SFEAT(I),I=1,NF
396: C
397: C      LONG LIST
398: 406  OUTPUT(2)'FEATURE VECTORS'
399:      WRITE(2,2314) I,I=1,10
400:      DO 410 J=1,NFV
401: C
402: C      LIST FEATURE VECTORS IF SS6 IS OFF
403:      IF(SENSE SWITCH 6) 411,409
404: 409  WRITE(2,2313) FEAT(I,J),I=1,20
405: 410  CONTINUE
406: 411  CONTINUE
407: C
408: C      SAVE UP TO 200 FEATURE VECTORS IN CORE
409: C      SAVE ONLY IF NEW FEATURE CARDS WERE READ THIS TIME
410:      IF(FLAG(1).EQ.0) GO TO 101
411:      NSAV=MIN(20*NFV,4000)
412:      DO 420 I=1,NSAV
413: 420  SAVE(I)=FEAT(I,1)
414:      FLAG(1)=0
415:      GO TO 101
416: C
417: C      OVERLAY NEW DATA
418: C
419: C      NFV PRINTS TO THE NEXT AVAILABLE SPACE IN ARRAY FEAT
420: 450  NFV=0
421:      GO TO 302
422: C
423: C      SECONDARY FEATURE CONTROL FOR BUS DETECTOR
424: C
425: 460  DECODE(76,2301,IN CARD) MADEP

```

```

426:      GO TO 302
427: C
428: C      TRADEOFF LISTING ON A FEATURE VALUE
429: C
430: 470  FLAG(4)=FLAG(4)+1
431: C
432: C      STORE ID NUMBER OF FEATURE TO BE TRADEOFF LISTED
433:      DECODE(76,2302,INCARD) ITV2(FLAG(4))
434:      GO TO 302
435: C
436: C      FEATURE TRANSFORMATION CONTROL USING PARAMETER CARDS
437: C
438: 480  DECODE(76,2302,INCARD) NPCRN
439:      OUTPUT(2) NPCRN
440:      GO TO 302
441: C
442: C*****
443: C      DATA PREPARATION MODULE
444: C*****
445: C
446: C      DETERMINE CLASS FROM TYPE, NUMBER OF SAMPLES PER CLASS AND
447: C      EXTRACT SELECTED FEATURES
448: 500  OUTPUT(2) ' ', 'PREP', ' '
449:      OUTPUT(3) 'PREP'
450:      DO 502 I=0,10
451: 502  NS(I)=0
452:      DO 520 K=1,NFV
453: C
454: C      DETERMINE CLASS NUMBER FROM TYPE
455: C      FEAT(3,K) = TYPE NUMBER, FEAT(10,K) = CLASS NUMBER OF VECTOR K
456: C      I=FEAT(10,K)-MIN(CLAS(FEAT(3,K)),NC)
457: C
458: C      EXTRACT SELECTED FEATURES
459: C      SFEAT(J) = JTH SELECTED FEATURE
460:      DO 510 J=1,NF
461: 510  FEAT(10+J,K)=FEAT(SFEAT(J)+10,K)
462: C
463: C      INCREMENT CLASS I SAMPLE COUNT
464: 520  NS(I)=NS(I)+1
465:      NSS=0
466:      DO 525 I=1,NC
467: 525  NSS=NSS+NS(I)
468: C
469: C      SUMMARY OUTPUT
470:      WRITE(2,2500)(I,I=0,10),(NS(I),I=0,NC)
471:      OUTPUT(2) ' ',NSS,' '
472:      IF(NSS.GT.0) GO TO 121
473:      OUTPUT(3) 'ERROR',NSS
474:      PAUSE 3
475:      GO TO 121
476: C
477: C      SECONDARY FEATURE TRANSFORMATION FOR BUS DETECTOR
478: C
479: 550  GO TO (551,551) MADE2
480: 551  J=1
481:      DO 554 K=1,NFV
482: C
483: C      PACK FEAT ARRAY IF NO 2ND CARD MATCHED TO THIS 1ST CARD DATA
484:      IF(FEAT(7,K).EQ.0) GO TO 554
485:      FACT=FEAT(11,K)+FFAT(12,K)
486:      IF(MADE2.EQ.2) FACT=FEAT(12,K)

```



```

487:      F16=(FEAT(11,K)*1000.)/FACT
488:      FEAT(11,J)=(FEAT(13,K)*1000.)/FACT
489:      FEAT(12,J)=(FEAT(14,K)*1000.)/FACT
490:      FACT=FEAT(13,K)+FEAT(14,K)
491:      FEAT(13,J)=(FEAT(16,K)*1000.)/FEAT(15,K)
492:      FEAT(14,J)=(FEAT(17,K)*1000.)/FEAT(15,K)
493:      FEAT(15,J)=FACT
494:      FEAT(16,J)=F16
495:      FEAT(17,J)=FEAT(11,J)+FEAT(12,J)
496:      DO 552 I=1,10
497: 552      FEAT(I,J) = FEAT(I,K)
498:      FEAT(4,J)=4
499:      J=J+1
500: 554      CONTINUE
501:      NFV=J-1
502:      WRITE(2,2308)
503:      OUTPUT(2) MODE2
504:      MODE2=0
505:      GO TO 404
506: C
507: C*****
508: C      STATISTICAL CALCULATIONS
509: C*****
510: C
511: C      CALCULATE CLASS MEANS, DISPERSIONS, COVARIANCES, CORRELATIONS,
512: C      STANDARD DEVIATIONS, MEAN-TO-CENTER DISTANCES,
513: C      BETWEEN CLASS COVARIANCE, STANDARD DEVIATION, PAIR-WISE DISTANCES
514: C      AND DIRECTIONAL COSINES AND ANGLES
515: C      TOTAL MEAN, COVARIANCE, CORRELATIONS, STANDARD DEVIATIONS.
516: C
517: 600      WRITE(2,2307) DATE,ID,ID1
518:      OUTPUT(2) ' ', 'STAT', ' '
519:      OUTPUT(3) 'STAT'
520:      DO 605 I=1,2630
521: 605      XBAR(I,J)=0.
522:      DO 620 L=1,NFV
523:      KL=FEAT(10,L)
524: C
525: C      CHECK FOR FEATURE VECTOR ASSIGNED TO CLASS ZERO
526: C      IF(KL.EQ.0) GO TO 620
527: C      IF(NS(KL).EQ.0) GO TO 620
528: C
529: C      CALCULATE CLASS WEIGHT FACTOR FROM NO. OF SAMPLES IN CLASS
530: C      P=1./NS(KL)
531: C
532: C      CHANGE FEATURE VECTOR TO FLOATING POINT
533: C      DO 610 M=1,NF
534: 610      TV1(M)=FLOAT(FEAT(10+M,L))
535: C
536: C      CALCULATE CLASS MEAN VECTORS
537: C      CALL VSUM(1,XBAR(1,KL),P,TV1, XBAR(1,KL),NF)
538: C
539: C      CALCULATE DISPERSION MATRICES
540: C      CALL VBERUD( TV1,TV1, TV1,NF)
541: C      CALL MSUM(1,DISP(1,1,KL),P,TV1,DISP(1,1,KL),NF)
542: 620      CONTINUE
543: C
544: C      LIST MEAN VECTORS
545: C      OUTPUT(2) ' ', 'MEAN VECTORS', ' '
546: C      WRITE(2,2601) I,I=1,NF
547: C      DO 630 I=1,NF

```

```

548: 630 WRITE(2,2601) (XBAR(I,J),J=1,NF)
549: C
550: C OPTION 5 TO LIST DISPERSION MATRICIES
551: IF(LIST(5).EQ.0) GO TO 644
552: OUTPUT(2) = 'DISPERSIONS', ' '
553: DO 640 K=1,NC
554: WRITE(2,2600) K
555: DO 640 I=1,NF
556: WRITE(2,2601) (DISP(I,J,K),J=1,NF)
557: 640 CONTINUE
558: 644 DO 645 I=1,200
559: 645 TM1(I,1)=0
560: DO 650 K=1,NC
561: C
562: C MEAN BUTER PRODUCT TO TM2
563: CALL VOPROD(XBAR(1,K),XBAR(1,K),TM2,NF)
564: C
565: C COVARIANCES TO C9V
566: CALL MSUM(1.,DISP(1,1,K),-1.,TM2,C9V(1,1,K),NF)
567: C
568: C ACCUMULATE TOTAL MEAN IN XBART
569: CALL VSUM(1.,XBART,PROB(K),XBAR(1,K),XBART,NF)
570: C
571: C SUM OF MEAN BUTER PRODUCTS TO TM1
572: CALL MSUM(1.,TM1,PROB(K),TM2,TM1,NF)
573: C
574: C ACCUMULATE WITHIN COVARIANCE IN WCOV
575: CALL MSUM(1.,WCOV,PROB(K),C9V(1,1,K),WCOV,NF)
576: 650 CONTINUE
577: C
578: C BETWEEN COVARIANCE TO BCOV
579: CALL VOPROD(XBART,XBART,TM2,NF)
580: CALL MSUM(1.,TM1,-1.,TM2,BCOV,NF)
581: C
582: C TOTAL COVARIANCE TO TCOV
583: CALL MSUM(1.,WCOV,1.,BCOV,TCOV,NF)
584: C
585: C TEST ONLY CHECK ON COVARIANCE *****
586: DO 661 K=1,100
587: 661 TM1(K,1)=0.
588: DO 662 K=1,NC
589: CALL MSUM(1.,TM1,PROB(K),DISP(1,1,K),TM1,NF)
590: 662 CONTINUE
591: CALL MSUM(1.,TM1,-1.,TM2,TM1,NF)
592: C
593: C OPTION 6 TO LIST XBART AND CLASS COVARIANCES
594: IF(LIST(6).EQ.0) GO TO 670
595: C
596: C LIST TOTAL MEAN VECTOR
597: WRITE(2,2602) NF,(I,I=1,NF),(XBART(I),I=1,NF)
598: C
599: C LIST CLASS COVARIANCES
600: OUTPUT(2) = 'COVARIANCES', ' '
601: DO 670 K=1,NC
602: WRITE(2,2600) K
603: DO 670 I=1,NF
604: WRITE(2,2601) (COV(I,J,K),J=1,NF)
605: 670 CONTINUE
606: C
607: C OPTION 7 TO LIST TCOV,WCOV,BCOV
608: IF(LIST(7).EQ.0) GO TO 678

```

```

609: C
610: C LIST TOTAL, BETWEEN AND WITHIN COVARIANCES
611: OUTPUT(2) ' ', 'TOTAL COVARIANCE', ' '
612: DO 672 I=1,NF
613: WRITE(2,2601) (TCOV(I,J),J=1,NF)
614: 672 CONTINUE
615: OUTPUT(2) ' ', 'WITHIN COVARIANCE', ' '
616: DO 674 I=1,NF
617: WRITE(2,2601) (WCOV(I,J),J=1,NF)
618: 674 CONTINUE
619: OUTPUT(2) ' ', 'BETWEEN COVARIANCE', ' '
620: DO 676 I=1,NF
621: WRITE(2,2601) (BCOV(I,J),J=1,NF)
622: 676 CONTINUE
623: C
624: C LIST TEST ONLY COVARIANCE CHECK
625: OUTPUT(2) ' ', 'COVARIANCE CHECK', ' '
626: DO 678 I=1,NF
627: WRITE(2,2601) (TM1(I,J),J=1,NF)
628: 678 CONTINUE
629: C
630: C CALCULATE CORRELATIONS
631: C
632: C CALCULATE AND LIST CLASS CORRELATIONS
633: DO 680 K=1,NC
634: DO 679 I=1,NF
635: TV1(I)=SQRT(COV(I,I,K))
636: 679 TV2(I)=1./TV1(I)
637: CALL DMDPRD(TV2,COV(1,1,K),TV2,T42,NF)
638: WRITE(2,2600) K
639: OUTPUT(2) 'STANDARD DEVIATIONS'
640: WRITE(2,2601) (TV1(I),I=1,NF)
641: OUTPUT(2) ' ', 'CORRELATIONS'
642: DO 680 I=1,NF
643: 680 WRITE(2,2601) (TM2(I,J),J=1,NF)
644: DO 681 I=1,NF
645: TM1(I,1)=SQRT(TCOV(I,I))
646: TM1(I,2)=SQRT(WCOV(I,I))
647: 681 TM1(I,3)=SQRT(BCOV(I,I))
648: DO 682 I=1,NF
649: DO 682 J=1,3
650: 682 TM1(I,J+3)=1./TM1(I,J)
651: C
652: C CALCULATE TOTAL CORRELATION AND LIST
653: CALL DMDPRD(TM1(1,4),TCOV,TM1(1,4),TM2,NF)
654: OUTPUT(2) ' ', 'TOTAL'
655: OUTPUT(2) 'STANDARD DEVIATIONS'
656: WRITE(2,2601) (TM1(I,1),I=1,NF)
657: OUTPUT(2) ' ', 'CORRELATIONS'
658: DO 684 I=1,NF
659: WRITE(2,2601) (TM2(I,J),J=1,NF)
660: 684 CONTINUE
661: C
662: C CALCULATE WITHIN CORRELATION AND LIST
663: CALL DMDPRD(TM1(1,5),WCOV,TM1(1,5),TM2,NF)
664: OUTPUT(2) ' ', 'WITHIN'
665: OUTPUT(2) 'STANDARD DEVIATIONS'
666: WRITE(2,2601) (TM1(I,2),I=1,NF)
667: OUTPUT(2) ' ', 'CORRELATIONS'
668: DO 686 I=1,NF
669: WRITE(2,2601) (TM2(I,J),J=1,NF)

```

```

670: 686 CONTINUE
671: C
672: C CALCULATE BETWEEN CORRELATIONS AND LIST
673: CALL DMOPRBD(TM1(1,6),BCBV,TM1(1,6),TM2,NF)
674: OUTPUT(2) ' ', 'BETWEEN'
675: OUTPUT(2) 'STANDARD DEVIATIONS'
676: WRITE(2,2601) (TM1(I,3),I=1,NF)
677: OUTPUT(2) ' ', 'CORRELATIONS'
678: DO 688 I=1,NF
679: WRITE(2,2601) (TM2(I,J),J=1,NF)
680: 688 CONTINUE
681: C
682: C CALCULATE AND LIST DISTANCE MATRIX
683: C
684: DO 690 K=1,NC
685: C
686: C MEANS RELATIVE TO TOTAL MEAN
687: CALL VSUM(1,XBAR(1,K),-1,XBART, TM2(1,K),NF)
688: 690 CONTINUE
689: C
690: C CALCULATE DISTANCE MATRIX
691: DO 692 I=1,NC
692: DO 692 J=1,I
693: CALL VIPRBD(TM2(1,I), TM2(1,J),TM1(I,J),NF)
694: TM1(J,I)=TM1(I,J)
695: 692 CONTINUE
696: C
697: C NORMALIZE
698: DO 694 I=1,NC
699: TV1(I)=SQRT(TM1(I,I))
700: 694 TV2(I)=1./TV1(I)
701: CALL DMOPRBD(TV2,TM1,TV2,TM2,NC)
702: C
703: C CALCULATE AND LIST PAIRWISE DISTANCES
704: DO 702 I=2,NC
705: DO 702 J=1,I-1
706: TM1(I,J)=SQRT(TM1(I,I)+TM1(J,J)-2.*TV1(I)*TV1(J)*TM2(I,J))
707: 702 TM1(J,I)=TM1(I,J)
708: OUTPUT(2) ' ', 'PAIRWISE DISTANCES'
709: WRITE(2,2600) I,I=1,NC
710: DO 704 I=1,NC
711: WRITE(2,2601) (TM1(I,J),J=1,NC)
712: 704 CONTINUE
713: C
714: C LIST DISTANCES AND COSINES
715: OUTPUT(2) ' ', 'CENTER TO MEAN', 'DISTANCE'
716: WRITE(2,2600) I,I=1,NC
717: WRITE(2,2601) TV1(I),I=1,NC
718: OUTPUT(2) ' ', 'DIRECTION COSINES'
719: WRITE(2,2600) I,I=1,NC
720: DO 706 I=1,NC
721: WRITE(2,2601) (TM2(I,J),J=1,NC)
722: 706 CONTINUE
723: C
724: C CONVERT TO DEGREES AND LIST
725: DO 708 I=2,NC
726: DO 708 J=1,I-1
727: COS = TM2(I,J)
728: SIN = SQRT(1-COS**2)
729: TM2(I,J) = ATAN(SIN,COS)*57.2958
730: TM2(J,I)=TM2(I,J)

```

```

731: 708 TM2(J,J)=0.
732: TM2(NC,NC)=0.
733: OUTPUT(2) ' ', 'DIRECTION ANGLES'
734: WRITE(2,2600) I,I=1,NC
735: DO 710 I=1,NC
736: WRITE(2,2601) (TM2(I,J),J=1,NC)
737: 710 CONTINUE
738: GO TO 141
739: C
740: C*****
741: C METHOD 1 CLASSIFIER
742: C*****
743: C
744: C GENERATES THE COEFFICIENT VECTORS FOR ALL CLASS PAIRS
745: C 1.LE.I<LT.J.LE.NC . THE VECTOR FOR I,J IS STORED IN
746: C METHOD 1 METHOD 2
747: C COV(1,I,J) COV(1,J,I) COEFFICIENT VECTOR
748: C COV(I,J,J) COV(J,I,I) CONSTANT
749: C
750: 800 WRITE(2,2801) KMODE
751: WRITE(3,2801) KMODE
752: FLAG(2)=KMODE
753: DO 810 I=1,NC-1
754: DO 810 J=I+1,NC
755: C
756: C CALCULATE PAIRWISE CLASS WEIGHT FACTORS PI AND PJ
757: C PI=PR08(I)
758: C PJ=PR08(J)
759: C PP=PI+PJ
760: C PI=PI/PP
761: C PJ=PJ/PP
762: C PIJ=PI*PJ
763: C QIJ=PIJ*(J-I)
764: C Y = I*PI+J*PJ
765: C
766: C PAIRWISE MEAN IN XBART, DIFFERENCE IN TV1
767: C CALL VSUM(PI,XBAR(1,I),PJ,XBAR(1,J),XBART,NF)
768: C CALL VSUM(QIJ,XBAR(1,J),-QIJ,XBAR(1,I),TV1,NF)
769: C
770: C KMODE = 1 FOR METHOD 1 AND 2 FOR METHOD 2
771: C GO TO (805,820) KMODE
772: C
773: C METHOD 1 *****
774: C FINDS THE LEAST SQUARES APPROXIMATION TO A LINEAR FUNCTION
775: C ASSIGNING I TO SAMPLES IN CLASS I, J TO SAMPLES IN CLASS J
776: C PAIRWISE TOTAL COVARIANCE TO TCOV
777: C
778: 805 CALL MSUM(PI,DISP(1,1,I),PJ,DISP(1,1,J),TCOV,NF)
779: CALL V0PR0D(XBART,XBART,TM1,NF)
780: CALL MSUM(1.,TCOV,-1.,TM1,TCOV,NF)
781: C
782: C CALCULATE INVERSE TO TCOV
783: C CALL T0INVR(K1,K2,NF,NF,TCOV,10,ITV,DET)
784: C OUTPUT(2) 'INVERSE',K1,K2,DET
785: C
786: C CALCULATE COEFFICIENT VECTOR TO COV(1,I,J)
787: C CALL MVPR0D(TCOV,TV1,COV(1,I,J),NF)
788: C
789: C CALCULATE CONSTANT TO COV(1,J,J)
790: C CALL V1PR0D(COV(1,I,J),XBART,B,NF)
791: COV(I,J,J)=Y-B

```



```

792: WRITE(2,2800) I,J, COV(I,J,I), (COV(K,I,J),K=1,NF)
793: 810 CONTINUE
794: C
795: C RETURN TO CONTROL LOOP
796: GO TO 151
797: C
798: C METHOD 2 *****
799: C FINDS THE DIRECTION OF SMALLEST DEVIATION WITHIN CLASS PAIR
800: C BY FINDING SMALLEST EIGENVALUE OF THE WITHIN COVARIANCE MATRIX
801: C PAIRWISE WITHIN COVARIANCE TO TM1
802: C
803: 820 CALL VPRBD(XBAR(1,I),XBAR(1,I),TM2,NF)
804: CALL MSUM(PI,DISP(1,1,I),-PI,TM2,TM1,NF)
805: CALL MSUM(1,TM1,PJ,DISP(1,1,J),TM1,NF)
806: CALL VPRBD(XBAR(1,J),XBAR(1,J),TM2,NF)
807: CALL MSUM(1,TM1,-PJ,TM2,TM1,NF)
808: C
809: C FIND SMALLEST EIGENVECTOR TO SCORE
810: CALL SEVAL(TM1,SCORE,Z,TM2,ITV,NF)
811: CALL VIPRBD(SCORE,TM1,X,NF)
812: X=X/91J*(J-1)
813: C
814: C COEFFICIENT VECTOR TO COV(1,J,I)
815: DO 830 N=1,NF
816: 830 COV(N,J,I)=SCORE(N)/X
817: C
818: C CONSTANT TO COV(J,I,I)
819: CALL VIPRBD(COV(1,J,I),XBAR(1,I),NF)
820: COV(J,I,I)=Y*B
821: WRITE(2,2800) I,J,COV(J,I,I), (COV(K,J,I),K=1,NF)
822: GO TO 810
823: C
824: C LOGICAL MODE CLASSIFIER *****
825: C USES FEATURES 1-LGBUNT
826: C CALCULATES A BOOLEAN BIT STRING FOR EACH FEATURE VECTOR
827: C
828: 840 DO 844 N=1,NF
829: SCORE(N)=0
830: DO 842 L=1,LGBUNT
831: C
832: C SHIFT LEFT ONE BIT
833: SCORE(N)=2*SCORE(N)
834: C
835: C SET BIT=1 IF FEATURE .GT. 0, BIT=0 IF FEATURE .LE. 0
836: IF (FEAT(10+L,N).GT.0) SCORE(N)=SCORE(N)+1
837: 842 CONTINUE
838: 844 CONTINUE
839: WRITE(2,2307) DATE,ID,ID1
840: OUTPUT(2) LGBUNT
841: CALL TRADEOFF(NC,LGBUNT,NFV,FEAT,TM1,0,TM2,NS,SCORE,3)
842: C
843: C RETURN TO CONTROL LOOP
844: GO TO 160
845: C
846: C*****
847: C PROCESS COEFFICIENT CARDS
848: C CALCULATE ALL FUNCTIONS BEFORE REPLACING ORIGINAL FEATURE VALUES
849: C*****
850: C
851: 1100 WRITE(2,2102)
852: DO 1101 K=1,FLAG(3)

```

```

853: 1101 WRITE(2,2800) ITV(K),ITV(10+K),TV1(K),(TM1(J,K),J=1,NF)
854: C
855: C CHECK FOR FEATURE PROJECTION
856: IF(NPCBN.GT.0) GO TO 1110
857: C
858: C OLD FEATURES REPLACED BY NEW FEATURES DEFINED BY COEFF CARDS
859: DO 1105 N=1,NFV
860: DO 1103 K=1,FLAG(3)
861: Y=TV1(K)
862: DO 1102 J=1,NF
863: 1102 Y=Y+TM1(J,K)*FEAT(10+J,N)
864: 1103 TV3(K)=Y
865: DO 1104 K=1,FLAG(3)
866: IF(ITV(K).GT.10) GO TO 1104
867: FEAT(10+ITV(K),N)=TV3(K)*ITV(10+K)+C*5
868: 1104 CONTINUE
869: 1105 CONTINUE
870: GO TO 1124
871: C
872: C PROJECTS ALL FEATURE VECTORS ONTO THE SUBSPACE ( OR ITS ORTHOGONAL
873: C COMPLEMENT ) DETERMINED BY USING THE COEFFICIENT VECTORS
874: C AS A SPANNING SET FOR THE SUBSPACE.
875: C THIS DOES NOT USE THE 2ND OR 4TH FIELDS OF THE 2 CARD.
876: C THE 3RD FIELD OF THE 1ST 2 CARD SPECIFIES THE SCALING.
877: C
878: 1110 IF(NPCBN.GE.3) GO TO 1124
879: C
880: C SET NSPAN = NR. OF COEFF. CARDS
881: NSPAN = FLAG(3)
882: SCALE=ITV(11)
883: IF(SCALE.EQ.0) SCALE=1.
884: C
885: C CALCULATE PROJECTION MATRIX TO BCOV
886: C
887: C TRANSPOSE OF A IS IN TM1
888: C PUT A INTO TM2
889: C
890: CALL TRANSP(TM1,TM2,NF)
891: CALL MPRBD(TM2,TM1,TCRV,NSPAN,NF,NSPAN)
892: CALL TDINVR(K1,K2,NSPAN,NSPAN,TCRV,10,ITV,DET)
893: OUTPUT(2) 'INVERSE',K1,K2,DET
894: CALL MPRBD(TCRV,TM2,ACBV,NSPAN,NSPAN,NF)
895: CALL MPRBD(TM1,BCBV,BCBV,NF,NSPAN,NF)
896: C
897: C TRANSFORM FEATURE VECTORS
898: C IF NPCBN = 1 , PROJECT TO SUBSPACE
899: C IF NPCBN = 2 , PROJECT TO COMPLEMENT OF SUBSPACE
900: C
901: GO TO(1112,1114) NPCBN
902: 1112 X=0.
903: Y=1.
904: OUTPUT(2) 'SUBSPACE'
905: GO TO 1116
906: 1114 X=1.
907: Y=-1.
908: OUTPUT(2) 'COMPLEMENT'
909: 1116 DO 1122 N=1,NFV
910: DO 1118 I=1,NF
911: 1118 TV1(I)=FEAT(10+I,N)
912: CALL MVPRBD(BCBV,TV1,TV3,NF)
913: CALL VSUM(X,TV1,Y,TV3,TV1,NF)

```

```

914:      DO 1120 I=1,NF
915: 1120 FEAT(10+I,N) = SCALE*TV1(I) +0.5
916: 1122 CONTINUE
917: C
918: C      SET NR OF COEFF. VECTORS TO ZERO
919: 1124 FLAG(3)=0
920: C
921: C      OUTPUT FEATURE VECTORS AND RETURN TO CONTRL LOOP
922:      GO TO 406
923: C
924: C      EXECUTE TRADEOFF LISTING OPTIONS
925: C
926: 1200 DO 1210 M=1,FLAG(4)
927:      MQ = ITV2(M)
928:      WRITE(2,2307) DATE,ID,ID1
929:      WRITE(2,1220) MQ
930: 1220 FORMAT(X,$TRADEOFF FOR FEATURE NUMBER $,I2)
931:      MP = 2
932:      IF(MQ .LT. 0) MQ = -MQ; MP = 4
933:      IF(MQ .GT. 10) GO TO 1210
934:      DO 1202 I=1,NFV
935: 1202 SCORE(I) = FEAT(MQ + 10, I)
936:      CALL TRADEOFF(MQ,NF,NFV,FEAT,TV1,0.,TV2,NS,SCORE,MP)
937: 1210 CONTINUE
938:      FLAG(4)=0
939: C
940: C      RETURN TO CONTRL LOOP
941:      GO TO 140
942: C
943: C*****
944: 2101 FORMAT(X,$TRADEOFF FOR $,X,$PAIR $,2I3,/,X)
945: 2102 FORMAT(1H0,$,COEFFICIENT CARDS$,/,
946: 1 $ KT KP CONST COEFF*****$)
947: 2300 FORMAT(I3,19A4)
948: 2301 FORMAT(16I)
949: 2302 FORMAT(I)
950: 2303 FORMAT( 1,10F)
951: 2304 FORMAT( 1,20I)
952: 2305 FORMAT(10I6)
953: 2306 FORMAT(I4,2A4)
954: 2307 FORMAT(1H1,$DATE $,I4,$ ID $,2A4)
955: 2308 FORMAT(1H1)
956: 2309 FORMAT(X,10F6.4,/)
957: 2310 FORMAT(X,$TYPE $,20I2)
958: 2311 FORMAT(X,$CLAS $,20I2,/)
959: 2312 FORMAT(X,10I3,/)
960: 2313 FORMAT(X,I3,I8,2I3,2I4,I7,2X,$$,3I3,10I8)
961: 2314 FORMAT(X,$ NR ID TY NF$,24X,$ CL$,10I8)
962: 2315 FORMAT(2I,11G)
963: 2316 FORMAT(20X, I3, 19A4)
964: 2500 FORMAT(X,$SELECTED SAMPLES$,/,X,$CLASS $,11I4,/,X,$NUMBER $,11I4)
965: 2600 FORMAT(X,$CLASS$,10I12)
966: 2601 FORMAT(2X,10G12.4)
967: 2602 FORMAT(1H0,$OVERALL MEAN$,/,X,I12,/,10G12.4)
968: 2800 FORMAT(X,2I4,11G11.4)
969: 2801 FORMAT(/X,$METHOD $,I2,/,X)
970:      END

```

NAME	TYPE	CLASS	BTCL LBC	DEC WORDS	NAME	TYPE	CLASS	BTCL LBC	DEC WORDS	NAME	TYPE	CLASS	BTCL LBC	DEC WORDS
A	R	SCALAR	11602P	2	ATAN	R	SPR9G	INTRIN		B	R	SCALAR	11641P	2
BCBV	R	ARRAY	31431C	200	CLAS	I	ARRAY	1755-C	20	CBS	R	SCALAR	11615P	2
CBV	R	ARRAY	24671C	2000	DATE	I	SCALAR	11557P	1	DET	R	SCALAR	11637P	2
JISP	R	ARRAY	20751C	2000	DWOPRAC	I	SPR9G	EXTERN		F16	R	SCALAR	11607P	2
FACT	R	SCALAR	11605P	2	FEAT	I	ARRAY	07664C	4000	FLAG	I	ARRAY	11533P	20
FLGAT	R	SPR9G	INTRIN		I	I	SCALAR	11574P	1	ID	I	SCALAR	11570P	1
ID1	I	SCALAR	11571P	1	INCARD	I	ARRAY	07641C	19	IP	I	SCALAR	11575P	1
ITV	I	ARRAY	32011C	20	ITVP	I	ARRAY	32061C	10	J	I	SCALAR	11577P	1
JP	I	SCALAR	11576P	1	K	I	SCALAR	11601P	1	K1	I	SCALAR	11635P	1
K2	I	SCALAR	11636P	1	KL	I	SCALAR	11611P	1	K1BUE	I	SCALAR	11566P	1
L	I	SCALAR	11600P	1	LCQUAT	I	SCALAR	11634P	1	LIST	I	ARRAY	11507P	20
M	I	SCALAR	11614P	1	MIN	I	SPR9G	INTRIN		MMDEP	I	SCALAR	11572P	1
MP	I	SCALAR	11654P	1	MPPR9G	I	SPR9G	EXTERN		MG	I	SCALAR	11653P	1
MSUM	I	SPR9G	EXTERN		MVPR9G	I	SPR9G	EXTERN		N	I	SCALAR	11647P	1
NC	I	SCALAR	11563P	1	NCT	I	SCALAR	11561P	1	NDT	I	SCALAR	11560P	1
NF	I	SCALAR	11565P	1	NFV	I	SCALAR	11562P	1	NPCBN	I	SCALAR	11567P	1
NS	I	ARRAY	17606C	11	NSAV	I	SCALAR	07640C	1	NSPAN	I	SCALAR	11650P	1
NSS	I	SCALAR	11573P	1	NT	I	SCALAR	11564P	1	P	R	SCALAR	11612P	2
PI	R	SCALAR	11621P	2	PIJ	R	SCALAR	11627P	2	PJ	R	SCALAR	11623P	2
PP	R	SCALAR	11625P	2	PRR9	R	ARRAY	17524C	20	PIJ	R	SCALAR	11631P	2
NESTART	I	SPR9G	EXTERN		SAVE	I	ARRAY	00000C	4000	SCALE	R	SCALAR	11651P	2
SCORE	R	ARRAY	32073C	400	SEVAL	I	SPR9G	EXTERN		SFEAT	I	ARRAY	17574C	10
SIN	R	SCALAR	11617P	2	SPRT	R	SPR9G	INTRIN		TCBV	R	ARRAY	37611C	200
TDIRVR	I	SPR9G	EXTERN		TM	R	ARRAY	20131C	200	TM2	R	ARRAY	27441C	200
TRADEOFF	I	SPR9G	EXTERN		TRANSP	R	SPR9G	EXTERN		TV1	R	ARRAY	31765C	20
TV2	R	ARRAY	32011C	20	TV2	R	ARRAY	32035C	20	VIPR9G	I	SPR9G	EXTERN	
VBR9G	I	SPR9G	EXTERN		VSUM	I	SPR9G	EXTERN		WCHV	R	ARRAY	31121C	200
X	R	SCALAR	11645P	2	XRAR	R	ARRAY	17621C	200	XBAKT	R	ARRAY	31741C	20
Y	R	SCALAR	11633P	2	Z	R	SCALAR	11643P	2					

LABEL	BTCL LBC	LABEL	BTCL LBC	LABEL	BTCL LBC	LABEL	BTCL LBC	LABEL	BTCL LBC	LABEL	BTCL LBC	LABEL	BTCL LBC
10	00054	12	00041	100	00062	101	00063	121	00072	140	00076	141	00076
141	00076	151	00111	152	00164	153	00231	155	00275	160	00303	300	00306
300	00306	302	00327	310	00454	320	01100	330	01166	331	01240	332	01262
332	01262	340	01275	350	01333	360	01371	370	01431	380	01456	382	01510
382	01510	384	01517	388	01522	390	01526	392	01556	394	01571	396	01601
396	01601	400	01622	401	01676	402	01702	404	01704	406	02056	409	02117
409	02117	410	02141	411	02144	420	02165	450	02176	460	02201	470	02211
470	02211	480	02223	500	02246	502	02274	510	02337	520	02361	525	02400
525	02400	550	02474	551	02501	552	02746	554	02777	600	03027	605	03067
605	03067	610	03146	620	03245	630	03311	640	03434	644	03442	645	03446
645	03446	650	03632	661	03667	662	03737	670	04102	672	04164	674	04240
674	04240	676	04315	678	04371	679	04425	680	04553	681	04651	682	04706
682	04706	684	05051	686	05177	688	05326	690	05365	692	05472	694	05524
694	05524	702	05634	704	05747	706	06113	708	06212	710	06327	800	06333
800	06333	805	06540	810	06775	820	07004	830	07171	840	07323	842	07400
842	07400	844	07403	1100	07450	1101	07464	1102	07570	1103	07616	1104	07665
1104	07665	1105	07677	1110	07674	1112	10015	1114	10033	1116	10050	1118	10070
1118	10070	1120	10137	1122	10161	1124	10164	1200	10167	1202	10261	1210	10316
1210	10316	1220	10222	2101	10324	2102	10337	2300	10365	2301	10371	2302	10374
2302	10374	2303	10376	2304	10401	2305	10404	2306	10407	2307	10412	2308	10422
2308	10422	2309	10425	2310	10432	2311	10437	2312	10445	2313	10451	2314	10463
2314	10463	2315	10476	2316	10501	2500	10506	2600	10526	2601	10534	2602	10540
2602	10540	2800	10553	2801	10560	3100	00527	3110	00623	3112	00673		

3114 00753      3115 00766      3116 01015      3120 01037      3123 01043

LOCAL VARIABLES (102 WORDS):

11507 LIST	11533 FLAG	11557 DATE	11560 NDT	11561 NCT	11562 NFV
11563 NC	11564 NT	11565 NF	11566 NMDE	11567 NPCBN	11570 ID
11571 ID1	11572 MDEP	11573 ASS	11574 I	11575 IP	11576 JP
11577 J	11600 L	11601 K	11602 A	11604 LCOUNT	11605 FACT
11607 F16	11611 KL	11612 P	11614 M	11615 CHS	11617 SIN
11621 PI	11623 PJ	11625 PP	11627 PIJ	11631 QIJ	11633 Y
11635 K1	11636 K2	11637 DET	11641 B	11643 Z	11645 X
11647 N	11650 NSPAN	11651 SCALE	11653 MQ	11654 MP	

BLANK COMMON (13771 WORDS):

00000 SAVE	07640 NSAV	07641 INCARD	07664 FEAT	17524 PRBB	17550 CLAS
17574 SFEAT	17600 NS	17621 XBAR	20131 TM1	20441 TM2	20751 DISP
24671 CBV	30611 TCPV	31121 WCBV	31431 BCBV	31741 XBART	31765 TV1
32011 ITV	32011 TV2	32035 TV3	32061 ITV2	32073 SCORE	

INTRINSIC SUBPROGRAMS USED:

ATAN      FL9AT      MIN      SQRT

EXTERNAL SUBPROGRAMS REQUIRED:

DMDPRD    MPRD    MSUM    MVRPD    RESTART    SEVAL    TDINVR    TRADEOFF  
TRANSP    VIPRD    VPRPD    VSUM

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	TOTAL WORDS	
GENERATED CODE:	4522	11326	
CONSTANTS:	36	00044	
TEMPS:	77	00115	
LOCAL VARIABLES:	102	00146	
TOTAL PROGRAM:	5037	11655	(PLUS BLANK COMMON)

END OF COMPILATION



```

1:      SUBROUTINE SEVAL(A,EV,LAMBDA,B,IV,N)
2:      C
3:      DIMENSION A(10,10),B(10,10),IV(10),EV(10)
4:      C
5:      C   A IS THE N BY N SYMMETRIC INPUT MATRIX
6:      C   B IS AN N BY N WORKING ARRAY
7:      C   IV IS AN INTEGER WORKING ARRAY OF N ELEMENTS
8:      C   EV (REAL) CONTAINS THE EIGENVECTOR ON EXIT
9:      C
10:     REAL NEWLAM,LAMBDA
11:     C
12:     C   INITIALIZE LAMBDA = 1/ ( TRACE OF INVERSE OF A )
13:     DO 10 I=1,N
14:     DO 10 J=1,N
15:     10  B(I,J)=A(I,J)
16:     CALL TDINVR(K1,K2,N,N,B,10,IV,DET)
17:     T=0.
18:     DO 20 I=1,N
19:     20  T=T+B(I,I)
20:     LAMBDA=1./T
21:     C
22:     C   GENERATE UP TO TEN TERMS IN THE SEQUENCE
23:     C   EV(1..N-1) = ( (A(1..N-1,1..N-1)-LAMBDA*I) INVERSE ) * A(1..N-1,N)
24:     C   EV(N) = 1
25:     C   LAMBDA = (EV' * A * EV) / (EV' * EV)
26:     C
27:     DO 40 K=1,10
28:     C
29:     C   CALCULATE EV
30:     DO 30 I=1,N-1
31:     B(I,I)=LAMBDA-A(I,I)
32:     DO 30 J=I+1,N-1
33:     B(I,J)=B(J,I)=-A(I,J)
34:     30  CONTINUE
35:     CALL TDINVR(K1,K2,N-1,N-1,B,10,IV,DET)
36:     CALL MVPR6D(B,A(1,N),EV,N-1)
37:     EV(N)=1.
38:     C
39:     C   CALCULATE LAMBDA
40:     CALL VIPR6D(EV,EV,VMAG,N)
41:     CALL MVPR6D(A,EV,B,N)
42:     CALL VIPR6D(EV,B,NEWLAM,N)
43:     NEWLAM=NEWLAM/VMAG
44:     C
45:     C   IF DONE, A(N,1..N)*EV = LAMBDA*EV(N)
46:     CALL VIPR6D(A(1,N),EV,T,N-1)
47:     T=ABS(NEWLAM-T-A(N,N))
48:     IF(T.LT.10.E-8) GO TO 50
49:     LAMBDA=NEWLAM
50:     40  CONTINUE
51:     50  OUTPUT(3) 'SEVAL NOT CONVERGING'
52:     50  RETURN
53:     END

```

NAME	TYPE	CLASS	BCIAL L9C	DEC WORDS
A	R	ARRAY	*00003F	DUMMY
DET	R	SCALAR	00535P	2
IV	I	ARRAY	*00007F	DUMMY
K1	I	SCALAR	00533P	1
MVPRBO	SPRG	EXTERN		
SEVAL	R	SCALAR	00525P	2
TDINVR	SPRG	EXTERN		

NAME	TYPE	CLASS	BCIAL L9C	DEC WORDS
ABS	R	SPRG	INTRIN	
EV	R	ARRAY	*00004P	DUMMY
J	I	SCALAR	00532P	1
KP	I	SCALAR	00534P	1
N	I	SCALAR	*00010P	DUMMY
SEVAL	SPRG	00000P		
VIPRBO	SPRG	EXTERN		

NAME	TYPE	CLASS	BCIAL L9C	DEC WORDS
R	R	ARRAY	*00006P	DUMMY
I	I	SCALAR	00531P	1
K	I	SCALAR	00541P	1
LAMBDA	R	SCALAR	*00005P	DUMMY
NEWLAM	R	SCALAR	00527P	2
T	R	SCALAR	00537P	2
VMAG	R	SCALAR	00542P	2

LABEL	BCIAL L9C	LABEL	BCIAL L9C	LABEL	BCIAL L9C	LABEL	BCIAL L9C	LABEL	BCIAL L9C	LABEL	BCIAL L9C
10	00046	20	00121	30	00225	40	00455	50	00474		

LOCAL VARIABLES (15 WORDS):

00525 SEVAL	00527 NEWLAM	00531 I	00532 J	00533 K1	00534 K2
00535 DET	00537 T	00541 K	00542 VMAG		

BLANK COMMON (0 WORDS)

ENTRY POINTS:

00000 SEVAL

INTRINSIC SUBPROGRAMS USED:

ABS

EXTERNAL SUBPROGRAMS REQUIRED:

MVPRBO TDINVR VIPRBO

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	BCIAL WORDS
GENERATED CODE:	317	00475
CONSTANTS:	13	00015
TEMPS:	11	00013
LOCAL VARIABLES:	15	00017
TOTAL PROGRAM:	356	00544

END OF COMPILATION

```

1:      SUBROUTINE TRADEOFF(NC,NF,NFV,FEAT,TV1,B,TV2,NS,SCORE,MODE)
2:  C
3:      DIMENSION TV1(10),TV2(10),FEAT(20,200),SCORE(200),NS(0:10)
4:      INTEGER FEAT
5:  C
6:  C      TABULAR OUTPUT OF TRADEOFF CURVE DATA
7:  C      FEAT VEC SCORES ARE RANKED AND FEAT VECs ORDERED BY SCORE
8:  C      FEAT = FEATURE VECTORS
9:  C      TV1=COEFFICIENT VECTOR
10: C      B=CONSTANT TERM
11: C      TV2,NS,SCORE ARE TEMPORARIES
12: C      MODE = 1 CALCULATES SCORE  $Y = TV1 \cdot X + B$  FOR ALL FEATURE VECTORS X
13: C      MODE=2,3,4 ASSUMES SCORES ARE ALREADY CALCULATED
14: C
15: C      IN MODE=1,2 SCORES ARE OUTPUT IN DECIMAL
16: C      IN MODE=3 SCORES ARE OUTPUT IN OCTAL (INTEGER PART)
17: C      MODE=4 OUTPUTS FEAT 1 - 10 ORDERED BY SCORE
18: C      MODE=11-14 SUPPRESSES PRINTING
19: C
20:      LL = 1
21:      IF (MODE .GT. 10) MODE = MODE - 10, LL = 2
22:      GO TO (1, 11, 11, 11), MODE
23:  C
24:  C      LIST COEFFICIENTS
25:  1  WRITE(2,102) B,(TV1(I),I=1,NF)
26:  C
27:  C      CALCULATE SCORES
28:      DO 10 N=1,NFV
29:      DO 5 I=1,NF
30:  5  TV2(I)=FEAT(I+10,N)
31:      CALL VIPROD(TV1,TV2,Y,NF)
32:  10 SCORE(N)=Y+B
33:  C
34:  C      RANK THE SCORES
35:  11 DO 12 N=1,NFV
36:  12 FEAT(8,N)=0
37:      DO 20 N=1,NFV
38:  C
39:  C      SET FEAT(9,N) TO RANK OF FEATURE VECTOR N
40:      K=0
41:      DO 15 I=1,NFV
42:      R=SCORE(N)
43:  15 IF (SCORE(I).LE.R) K=K+1
44:  20 FEAT(9,N)=K
45:      DO 30 N=1,NFV
46:  C
47:  C      SET FEAT(8,N) TO LOCATION OF FEATURE VECTOR WITH RANK N
48:      K=FEAT(9,N)
49:  25 IF (FEAT(8,K).EQ.0) GO TO 30
50:      K=K-1
51:      GO TO 25
52:  30 FEAT(8,K)=N
53:  C
54:  C      ORDERED OUTPUT
55:  C
56:      IF (LL .EQ. 2) GO TO 99
57:      DO 35 I=0,10
58:  35 NS(I)=0
59:      GO TO (37, 37, 37, 35), MODE
60:  36 WRITE(2,104) I, I = 1, 10

```

```

61:      GO TO 38
62: 37   WRITE(2,100) I,I=0,NC
63: 38   DO 60 N=1,NFV
64:      J=FEAT(8,N)
65:      K=FEAT(10,J)
66:      NS(K)=NS(K)+1
67:      GO TO (40, 40, 45, 50), MODE
68: 40   WRITE(2,101) N,(FEAT(I,J),I=1,3),(FEAT(I,J),I=5,7),K,SCORE(J)
69:      1,K,NS(K)
70:      GO TO 60
71: 45   WRITE(2,103) N,(FEAT(I,J),I=1,3),(FEAT(I,J),I=5,7),K,(LL=SCORE(J))
72:      1,K,NS(K)
73:      GO TO 60
74: 50   WRITE(2, 105) N, FEAT(1, J), K, SCORE(J), (FEAT(1, J), I = 11, 20)
75: 60   CONTINUE
76: 99   RETURN
77: C
78: 100  FORMAT(7X,$N0      ID TY$,16X,$CL      SCORE  $,11I3)
79: 101  FORMAT(X,2I4,I8,I3,2I4,I7,I3,F14.4,N(3X),I3)
80: 102  FORMAT(X,$CBEFF$,/X,11F10.5)
81: 103  FORMAT(X,2I4,I8,I3,2I4,I7,I3,6X,08,N(3X),I3)
82: 104  FORMAT(7X,$N0 CL      SCORE  $, 10I6)
83: 105  FORMAT(X, 2I4, I3, F14.4, 10I6)
84:      END

```

NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS
----	----	----	-----	-----
B	R	SCALAR	*00010P	DUMMY
J	I	SCALAR	01030P	1
MBDE	I	SCALAR	*00014P	DUMMY
NF	I	SCALAR	*00004P	DUMMY
R	R	SCALAR	01026P	2
TRADE8FF	SPR8G		00000P	
VIPR8D	SPR8G	EXTERN		

NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS
----	----	----	-----	-----
FEAT	I	ARRAY	*00006P	DUMMY
K	I	SCALAR	01025P	1
N	I	SCALAR	01022P	1
NFV	I	SCALAR	*00005P	DUMMY
SCHRE	R	ARRAY	*00013P	DUMMY
TV1	R	ARRAY	*00007P	DUMMY
Y	R	SCALAR	01023P	2

NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS
----	----	----	-----	-----
I	I	SCALAR	01021P	1
LL	I	SCALAR	01020P	1
NC	I	SCALAR	*00003P	DUMMY
NS	I	ARRAY	*00012P	DUMMY
TRADE8FF	R	SCALAR	01016P	2
TV2	R	ARRAY	*00011P	DUMMY

LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC
----	-----	----	-----	----	-----	----	-----	----	-----
1	00052	5	0011F	10	00155	11	00167	12	00177
20	00256	25	00311	30	00325	35	00347	36	00364
38	00416	40	00457	45	00537	50	00622	60	00666
100	00672	101	00707	102	00722	103	00731	104	00744
								105	00755

LOCAL VARIABLES (11 WORDS):

01016 TRADE8FF	01020 LL	01021 I	01022 N	01023 Y	01025 K
01026 R	01030 J				

BLANK COMMON (0 WORDS)

ENTRY POINTS:

00000 TRADE8FF

EXTERNAL SUBPROGRAMS REQUIRED:

VIPR8D

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	BCIAL WORDS
----	-----	-----
GENERATED CODE:	501	00765
CONSTANTS:	15	00017
TEMPS:	10	00012
LOCAL VARIABLES:	11	00013
----	-----	-----
TOTAL PROGRAM:	537	01031

END OF COMPILATION



```

1:      SUBROUTINE VIPROD(A,B,C,N)
2:  C
3:      DIMENSION A(10),B(10)
4:      C=0.
5:      DO 10 I=1,N
6:  10   C=C+A(I)*B(I)
7:      RETURN
8:      END

```

NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS	NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS	NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS
----	----	----	-----	-----	----	----	----	-----	-----	----	----	----	-----	-----
A		R ARRAY	*00003P	DUMMY	B		R ARRAY	*00004P	DUMMY	C		R SCALAR	*00005P	DUMMY
I		I SCALAR	00053P	1	N		I SCALAR	*00006P	DUMMY	VIPROD		R SCALAR	00051P	2
VIPROD		SPRNG	00000P											

LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
10	00030										

LOCAL VARIABLES (3 WORDS):

00051 VIPROD      00053 I

BLANK COMMON (0 WORDS)

ENTRY POINTS:

00000 VIPROD

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	BCIAL WORDS
-----	-----	-----
GENERATED CODE:	36	00044
CONSTANTS:	3	00003
TEMPS:	2	00002
LOCAL VARIABLES:	3	00003
-----	-----	-----
TOTAL PROGRAM:	44	00054

END OF COMPILATION

```

1:      SUBROUTINE V9PRD(A,B,C,N)
2:  C
3:      DIMENSION A(10),B(10),C(10,10)
4:      DO 10 I=1,N
5:      DO 10 J=1,N
6:  10    C(I,J)=A(I)*B(J)
7:      RETURN
8:      END

```

NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS	NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS	NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS
A	R	ARRAY	*00003P	DUMMY	B	R	ARRAY	*00004P	DUMMY	C	R	ARRAY	*00005P	DUMMY
I	I	SCALAR	00106P	1	J	I	SCALAR	00107P	1	N	I	SCALAR	*00006P	DUMMY
V9PRD	R	SCALAR	00104P	2	V9PRD		SPRNG	00000P						

LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC
10	00041										

LOCAL VARIABLES (4 WORDS):

00104 V9PRD      00106 I      00107 J

BLANK COMMON (0 WORDS)

ENTRY POINTS:

00000 V9PRD

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	BCIAL WORDS
GENERATED CODE:	58	00072
CONSTANTS:	5	00005
TEMPER:	5	00005
LOCAL VARIABLES:	4	00004
TOTAL PROGRAM:	72	00110

END OF COMPILATION

```

1:      SUBROUTINE VSUM(S1,A,S2,B,C,N)
2:      C
3:      DIMENSION A(10),B(10),C(10)
4:      DO 10 I=1,N
5: 10    C(I)=S1*A(I)+S2*B(I)
6:      RETURN
7:      END

```

NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS	NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS	NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS
A	R	ARRAY	00004P	DUMMY	B	R	ARRAY	00006P	DUMMY	C	R	ARRAY	00007P	DUMMY
I	I	SCALAR	00063R	1	N	I	SCALAR	00010P	DUMMY	S1	R	SCALAR	00003P	DUMMY
S2	R	SCALAR	00005P	DUMMY	VSUM	R	SCALAR	00061P	2	VSUM	R	SCALAR	00000P	DUMMY

LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC
10	00033										

LOCAL VARIABLES (3 WORDS):

00061 VSUM 00063 I

BLANK COMMON (0 WORDS)

ENTRY POINTS:

00000 VSUM

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	BCIAL WORDS
GENERATED CODE:	42	00052
CONSTANTS:	3	00003
TEMPS:	4	00004
LOCAL VARIABLES:	3	00003
TOTAL PROGRAM:	52	00064

END OF COMPILATION

```

1:      SUBROUTINE MPRBD(A,B,C,N(L))
2:      C
3:      C      A IS N1*N2, B IS N2*N3, AND C IS RETURNED N1*N3
4:      DIMENSION A(10,10),B(10,10),C(10,10)
5:      N1=N(1)
6:      IF(L.EQ.3) GO TO 2
7:      C
8:      C      IF ONLY N1 OR N1 AND N2 SPECIFIED THEN N3 = N2 * N1
9:      N2=N3*N1
10:     GO TO 3
11:     2      N2=N(2)
12:     3      N3=N(3)
13:     3      CONTINUE
14:     20     I=1,N1
15:     20     J=1,N3
16:     C(I,J)=0.
17:     10     K=1,N2
18:     20     C(I,J)=A(I,K)*B(K,J)+C(I,J)
19:     20     CONTINUE
20:     RETURN
21:     END

```

NAME	TYPE	CLASS	BCTAL LOC	DEC WORDS	NAME	TYPE	CLASS	BCTAL LOC	DEC WORDS	NAME	TYPE	CLASS	BCTAL LOC	DEC WORDS
A	R	ARRAY	*00003P	DUMMY	B	R	ARRAY	*00004P	DUMMY	C	R	ARRAY	*00005P	DUMMY
I	I	SCALAR	00166P	1	J	I	SCALAR	00167P	1	K	I	SCALAR	00170P	1
L	I	SCALAR	00162P	1	MPRBD	I	SCALAR	00161P	1	MPRBD	I	SCALAR	00000P	1
N	I	MULOMY	*00006P	DUMMY	N1	I	SCALAR	00163P	1	N2	I	SCALAR	00164P	1
N3	I	SCALAR	00165P	1										

LABEL	BCTAL LOC	LABEL	BCTAL LOC	LABEL	BCTAL LOC	LABEL	BCTAL LOC	LABEL	BCTAL LOC	LABEL	BCTAL LOC
2	00034	3	00042	10	00103	20	00137				

LOCAL VARIABLES (8 WORDS):

00161 MPRBD	00162 L	00163 N1	00164 N2	00165 N3	00166 I
00167 J	00170 K				

BLANK COMMON (0 WORDS)

ENTRY POINTS:

00000 MPRBD

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	BCTAL WORDS
GENERATED CODE:	102	00146
CONSTANTS:	5	00005
TEMPS:	6	00006
LOCAL VARIABLES:	8	00010
TOTAL PROGRAM:	121	00171

END OF COMPILATION

```

1:      SUBROUTINE MVPRD(A,B,C,N)
2:  C
3:      DIMENSION A(10,10),B(10),C(10)
4:      DO 20 I=1,N
5:        C(I)=0*
6:        DO 10 J=1,N
7:          C(I)=C(I)+A(I,J)*B(J)
8:        20 CONTINUE
9:      RETURN
10:     END

```

NAME	TYPE	CLASS	BCIAL LOC	DEC WORDS	NAME	TYPE	CLASS	BCIAL LOC	DEC WORDS	NAME	TYPE	CLASS	BCIAL LOC	DEC WORDS
A	R	ARRAY	000003P	DUMMY	B	R	ARRAY	000004P	DUMMY	C	R	ARRAY	000005P	DUMMY
I	I	SCALAR	00114P	1	J	I	SCALAR	00115P	1	MVPRD	I	SCALAR	00113P	1
MVPRD		SPRNG	00000P		N	I	SCALAR	000006P	DUMMY					

LABEL	BCIAL LOC	LABEL	BCIAL LOC	LABEL	BCIAL LOC	LABEL	BCIAL LOC	LABEL	BCIAL LOC	LABEL	BCIAL LOC
10	00047	20	00075								

LOCAL VARIABLES (3 WORDS):

00113 MVPRD 00114 I 00115 J

BLANK COMMON (0 WORDS)

ENTRY POINTS:

00000 MVPRD

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	BCIAL WORDS
GENERATED CODE:	65	00101
CONSTANTS:	5	00005
TEMPS:	5	00005
LOCAL VARIABLES:	3	00003
TOTAL PROGRAM:	78	00116

END OF COMPILATION



```

1:      SUBROUTINE MSUM(S1,A,S2,B,C,N)
2:      C
3:      DIMENSION A(10,10),B(10,10),C(10,10)
4:      DO 20 I=1,N
5:      DO 10 J=1,N
6:      C(I,J)=S1*A(I,J)+S2*B(I,J)
7:      20 CONTINUE
8:      RETURN
9:      END

```

NAME	TYPE	CLASS	BCTAL LBC	DEC WORDS	NAME	TYPE	CLASS	BCTAL LBC	DEC WORDS	NAME	TYPE	CLASS	BCTAL LBC	DEC WORDS
A	R	ARRAY	00004P	DUMMY	B	R	ARRAY	00004P	DUMMY	C	R	ARRAY	00007P	DUMMY
I	I	SCALAR	00102P	1	J	I	SCALAR	00103P	1	MSUM	I	SCALAR	00101P	1
MSUM	SPRNG		00000P		N	I	SCALAR	00015P	DUMMY	S1	R	SCALAR	00003P	DUMMY
S2	R	SCALAR	00003P	DUMMY										

LABEL	BCTAL LBC	LABEL	BCTAL LBC	LABEL	BCTAL LBC	LABEL	BCTAL LBC	LABEL	BCTAL LBC	LABEL	BCTAL LBC
10	00043	20	00064								

LOCAL VARIABLES (3 WORDS):

00101 MSUM      00102 I      00103 J

BLANK COMMON (0 WORDS)

ENTRY POINTS:

00000 MSUM

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	BCTAL WORDS
GENERATED CODE:	56	00070
CONSTANTS:	4	00004
TEMPS:	5	00005
LOCAL VARIABLES:	3	00003
TOTAL PROGRAM:	68	00104

END OF COMPILATION

```

1:      SUBROUTINE DMDPRD(A,B,C,D,N)
2:  C
3:      DIMENSION A(10),B(10,10),C(10),D(10,10)
4:      DO 20 I=1,N
5:      DO 10 J=1,N
6:  10  D(I,J)=A(I)*B(I,J)*C(J)
7:      CONTINUE
8:      RETURN
9:      END

```

NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS	NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS	NAME	TYPE	CLASS	BCIAL LBC	DEC WORDS
A	R	ARRAY	*00003P	DUMMY	B	R	ARRAY	*00004P	DUMMY	C	R	ARRAY	*00005P	DUMMY
D	R	ARRAY	*00006P	DUMMY	DMDPRD	R	SCALAR	00112P	2	DMDPRD	R	SCALAR	00000P	0
I	I	SCALAR	00114P	1	J	I	SCALAR	00115P	1	N	I	SCALAR	*00007P	DUMMY

LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC	LABEL	BCIAL LBC
10	00045	20	00074								

LOCAL VARIABLES (4 WORDS):

00112 DMDPRD 00114 I 00115 J

BLANK COMMON (0 WORDS)

ENTRY POINTS:

00000 DMDPRD

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	BCIAL WORDS
GENERATED CODE:	64	00100
CONSTANTS:	5	00005
TEMPS:	5	00005
LOCAL VARIABLES:	4	00004
TOTAL PROGRAM:	78	00116

END OF COMPILATION

```

1:      SUBROUTINE TRANSP(A,B,N)
2:      C
3:      DIMENSION A(10,10),B(10,10)
4:      DO 20 I=1,N
5:      DO 10 J=1,N
6:      10 B(I,J)=A(J,I)
7:      20 CONTINUE
8:      RETURN
9:      END

```

NAME	TYPE	CLASS	BCTAL LOC	DEC WORDS	NAME	TYPE	CLASS	BCTAL LOC	DEC WORDS	NAME	TYPE	CLASS	BCTAL LOC	DEC WORDS
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
A	R	ARRAY	00003P	DUMMY	B	R	ARRAY	00004P	DUMMY	I	I	SCALAR	00076P	1
J	I	SCALAR	00077P	1	N	I	SCALAR	00005P	DUMMY	TRANSP	R	SCALAR	00074P	2
TRANSP		SPRNG	00000P											

LABEL	BCTAL LOC	LABEL	BCTAL LOC	LABEL	BCTAL LOC	LABEL	BCTAL LOC	LABEL	BCTAL LOC	LABEL	BCTAL LOC
----	----	----	----	----	----	----	----	----	----	----	----
10	00035	20	00067								

LOCAL VARIABLES (4 WORDS):

00074 TRANSP      00076 I      00077 J

BLANK COMMON (0 WORDS)

ENTRY POINTS:

00000 TRANSP

HIGHEST ERROR SEVERITY: 0 (NO ERRORS)

	DEC WORDS	BCTAL WORDS
----	----	----
GENERATED CODE:	52	00064
CONSTANTS:	4	00004
TEMPS:	4	00004
LOCAL VARIABLES:	4	00004
----	----	----
TOTAL PROGRAM:	64	00100

END OF COMPILATION

```

1:      SUBROUTINE TDINVR(ISOL,IDSOL,NR,NC,A,MRA,KWA,DET)
2:  C
3:  C      SUBROUTINE TO FIND THE INVERSE, SOLVE SETS OF SIMULTANEOUS
4:  C      EQUATIONS, AND/OR CALCULATE THE DETERMINANT OF A REAL MATRIX
5:  C
6:  C      ISOL = 1 IF INVERSE FOUND OR EQUATIONS SOLVED,
7:  C           = 2 IF UNABLE TO SOLVE, = 3 IF INPUT ERROR.
8:  C      IDSOL = 1 IF NO OVERFLOW, = 2 IF OVERFLOW.
9:  C      NR IS NUMBER OF ROWS OF A.
10: C      ABS(NC) IS NUMBER OF COLUMNS OF A. NO INVERSE FOUND IF NC IS NEG.
11: C      FIRST NR COLUMNS OF A FORM SQUARE MATRIX TO BE INVERTED ON INPUT,
12: C      RETURNS THE INVERSE ON EXIT.
13: C      NEXT ABS(NC)-NR COLUMNS OF A ARE CONSTANT VECTORS FOR
14: C      ABS(NC)-NR SYSTEMS OF EQUATIONS, RETURN SOLUTIONS ON EXIT.
15: C      MRA IS NUMBER OF ROWS RESERVED FOR A IN MAIN PROGRAM DIMENSION.
16: C      KWA IS WORKING ARRAY OF THE FORM KWA(NR).
17: C      KWA MAY BE A DUMMY VARIABLE IF NO EQUATIONS ARE TO BE SOLVED.
18: C      DET IS THE VALUE OF DETERMINANT IF ISOL = IDSOL = 1.
19: C
20: C      A, KWA TAKE THE DIMENSIONS GIVEN THEM IN THE CALLING PROGRAM
21: C
22:      DIMENSION A(1),KWA(1)
23:      IR=NR
24:      ISOL =1
25:      IDSOL=1
26:      IF(NR) 61,61,11
27: 11 IF(IR-MRA)12,12,61
28: 12 IC=ABS(NC)
29: IF(IC-IR) 13,14,14
30: 13 IC=IR
31: 14 IBMP=1
32:      JBMP=MRA
33:      KBMP=JBMP+IBMP
34:      NES=IR*JBMP
35:      NET=IC*JBMP
36:      IF(NC) 15,61,16
37: 15 MDIV=JBMP+1
38:      IRIC=IR-IC
39:      GO TO 17
40: 16 MDIV=1
41: 17 MAD =MDIV
42:      MSER=1
43:      KSER=IR
44:      MZ =1
45:      DET=1.0
46:  C
47:  C      MAIN LOOP*****
48:  C      ROW OPERATIONS ON MATRIX TO INVERT, SOLVE SYSTEMS, CALC. DET.
49:  C
50: 18 PIV=0.0
51:      I=MSER
52:  C
53:  C      FIND LARGEST ELEMENT IN COLUMN TO PIV
54: 19 IF(I=KSER) 20,20,23
55: 20 IF(ABS(A(I))-PIV)22,22,21
56: 21 PIV=ABS(A(I))
57:      IP=I
58: 22 I=I+IBMP
59:      GO TO 19
60: 23 IF(PIV) 24,62,24

```

```

61:      24 IF(NC) 26,25,25
62: C
63: C      INVERSE WANTED...ACCUMULATE INVERSE AND SAVE POINTER TO PIV
64:      25 I=IP-((IP-1)/JBMP)*JBMP
65:      J=MSER-((MSER-1)/JBMP)*JBMP
66:      JJ=MSER/KBMP+1
67:      II=JJ+(IP=MSER)
68:      K= A(JJ)=II
69:      GO TO 27
70:      26 I=IP
71:      J=MSER
72:      27 IF(IP=MSER) 61,31,28
73: C
74: C      INTERCHANGE ROWS TO GET PIVOT ELEMENT ON MAIN DIAGONAL
75:      28 IF(J=NET) 29,29,30
76:      29 PSTB=A(I)
77:      A(I)=A(J)
78:      A(J)=PSTB
79:      I=I+JBMP
80:      J=J+JBMP
81:      GO TO 28
82: C
83: C      ACCUMULATE DETERMINANT AND CHECK FOR OVERFLOW
84:      30 DET=-DET
85:      31 PSTB=A(MSER)
86:      DET=DET*PSTB
87:      CALL OVERFL(IVF)
88:      IF(IVF=1)63,63,35
89:      35 PSTB=1.0/PSTB
90:      CALL DVCHK(IVF)
91:      IF(IVF=1)63,63,136
92: 136 CONTINUE
93:      A(MSER)=1.0
94:      I=MDIV
95: C
96: C      DIVIDE PIVOT ROW BY PIVOT
97:      36 IF(I=NET) 37,37,39
98:      37 A(I)=A(I)*PSTB
99:      I=I+JBMP
100:     GO TO 36
101: C
102: C      USE PIVOT ROW TO ZERO PIVOT COLUMN ELEMENTS
103:      39 IF(MZ=KSER) 40,40,145
104:      40 IF(MZ=MSER) 41,44,41
105:      41 I=MAD
106:      J=MDIV
107:      PSTB=A(MZ)
108:      IF(PSTB) 142,44,142
109:      142 A(MZ)=0.0
110:      42 IF(J=NET) 43,43,44
111:      43 A(I)=A(I)-A(J)*PSTB
112:      J=J+JBMP
113:      I=I+JBMP
114:      GO TO 42
115:      44 MAD=MAD+JBMP
116:      MZ=MZ+JBMP
117:      GO TO 39
118:      145 KSER=KSER+JBMP
119: C
120: C      EXIT FROM MAIN LOOP
121:      IF(KSER=NES) 46,46,53

```



```

122:      46 MSER=MSER+KBMP
123:      IF(NC) 48,47,47
124:      47 MDIV=MDIV+IBMP
125:      MZ=((MSER-1)/JBMP)*JBMP+1
126:      MAD=1
127:      GO TO 52
128:      48 MDIV=MDIV+KBMP
129:      IF(IRIC) 50,49,50
130:      49 MZ=MSER+IBMP
131:      GO TO 51
132:      50 MZ=((MSER-1)/JBMP)*JBMP+1
133:      51 MAD=MZ+JBMP
134:      52 GO TO 18
135: C
136: C      END MAIN LOOP*****
137: C
138:      53 IF(NC) 65,54,54
139: C
140: C      BEGIN INTERCHANGE OF COLUMNS USING POINTERS TO PIVOTS
141:      54 JR=IR
142:      55 IF(JR) 61,65,56
143:      56 IF(KWA(JR)=JR) 61,60,57
144:      57 K=(JR-1)*JBMP
145:      J=K+IR
146:      L=(KWA(JR)-1)*JBMP+IR
147:      58 IF(J=K) 61,60,59
148:      59 PST0=A(L)
149:      A(L)=A(J)
150:      A(J)=PST0
151:      J=J-IBMP
152:      L=L-IBMP
153:      GO TO 58
154:      60 JR=JR-1
155:      GO TO 55
156:      61 IS0L=3
157:      GO TO 65
158:      62 DET=0.0
159:      IS0L=2
160:      IDS0L=1
161:      GO TO 65
162:      63 IS0L = 2
163:      IDS0L = 2
164:      65 RETURN
165:      END

```

NAME	TYPE	CLASS	BCIAL LOC	DEC WORDS
----	----	----	----	----
A	R	ARRAY	*00007P	DUMMY
DVCHK		SPRGG	EXTERN	
IBMP	I	SCALAR	00666P	1
II	I	SCALAR	00707P	1
IRIC	I	SCALAR	00674P	1
J	I	SCALAR	00705P	1
JR	I	SCALAR	00713P	1
KSER	I	SCALAR	00677P	1
MAJ	I	SCALAR	00676P	1
MSER	I	SCALAR	00676P	1
NES	I	SCALAR	00671P	1
OVERFL		SPRGG	EXTERN	
TDINVR	R	SCALAR	00662P	2

NAME	TYPE	CLASS	BCIAL LOC	DEC WORDS
----	----	----	----	----
ARS	R	SPRGG	INTRIN	
I	I	SCALAR	00703P	1
IC	I	SCALAR	00666P	1
IP	I	SCALAR	00704P	1
ISHL	I	SCALAR	*00003P	DUMMY
JRMP	I	SCALAR	00667P	1
K	I	SCALAR	00714P	1
KWA	I	ARRAY	*00011P	DUMMY
MDIV	I	SCALAR	00675P	1
M7	I	SCALAR	00700P	1
NET	I	SCALAR	00670P	1
PIV	R	SCALAR	00711P	2
TDINVR	R	SPRGG	00000P	

NAME	TYPE	CLASS	BCIAL LOC	DEC WORDS
----	----	----	----	----
DET	R	SCALAR	*00012P	DUMMY
IABS	I	SPRGG	INTRIN	
IDSOL	I	SCALAR	*00004P	DUMMY
IR	I	SCALAR	00664P	1
IvF	I	SCALAR	00712P	1
JJ	I	SCALAR	00706P	1
KBMP	I	SCALAR	00670P	1
L	I	SCALAR	00715P	1
MMA	I	SCALAR	*00010P	DUMMY
NC	I	SCALAR	*00006P	DUMMY
NR	I	SCALAR	*00005P	DUMMY
PSTH	R	SCALAR	00710P	2

LABEL	BCIAL LOC	LABEL	BCIAL LOC	LABEL	BCIAL LOC	LABEL	BCIAL LOC	LABEL	BCIAL LOC	LABEL	BCIAL LOC
----	----	----	----	----	----	----	----	----	----	----	----
11	00033	12	00037	13	00047	14	00051	15	00075	16	00104
17	00106	18	00120	19	00124	20	00130	21	00142	22	00154
23	00157	24	00162	25	00165	26	00224	27	00230	28	00236
29	00242	30	00275	31	00300	35	00320	36	00342	37	00346
39	00360	40	00364	41	00370	42	00413	43	00417	44	00441
46	00454	47	00461	48	00474	49	00503	50	00507	51	00517
52	00522	53	00523	54	00526	55	00530	56	00535	57	00544
58	00563	59	00571	60	00626	61	00631	62	00634	63	00643
65	00647	136	00332	142	00425	145	00446				

#### LOCAL VARIABLES (28 WORDS):

00662 TDINVR	00664 IR	00665 IC	00666 IBMP	00667 JBMP	00670 KBMP
00671 NES	00672 NET	00673 MDIV	00674 IRIC	00675 MAJ	00676 MSER
00677 KSER	00700 M7	00701 PIV	00703 I	00704 IP	00705 J
00706 JJ	00707 II	00710 PSTH	00712 IvF	00713 JR	00714 K
00715 L					

#### BLANK CRYSTAL (0 WORDS)

#### ENTRY POINTS:

00000 TDINVR

#### INTRINSIC SUBPROGRAMS USED:

ABS IABS

#### EXTERNAL SUBPROGRAMS REQUIRED:

DVCHK OVERFL

#### HIGHEST ERROR SEVERITY: (NO ERRORS)

	DEC	BCIAL
	WORDS	WORDS
----	----	----
GENERATED CODE:	424	00650
CONSTANTS:	8	00010
TEMPORARY:	2	00002
LOCAL VARIABLES:	28	00034
----	----	----
TOTAL PROGRAM:	462	00716

END OF COMPILATION  
 ΔLOAD MAP,ULIB,FLIB,XM

B=00

\*\*\*\* MAJOR ERRORS \*\*\*\*

NONE

\*\*\*\* MINOR ERRORS \*\*\*\*

NONE

\*\*\*\* MEMORY MAP \*\*\*\*

LABEL	ENTRY	ORIGIN	SIZE	ERRORS
8E0FEXIT	51241	51174		
810TRIG	51240	51174		
8E0FFLAG	51237	51174		
810FLAG	51235	51174		
8BUFFLAG	51234	51174		
9R0VTRIG	51216	51174		
9INITIAL	51174	51174		
7SERR0R	51246	51246		
70CTBCD	51427	51366		
9ERR0R	51366	51366		
7UNITADR	51472	51472		
9ST0P	51517	51517		
7SNSERR	51531	51531		
9IFSWICH	51560	51560		
9PAUSE	51611	51611		
9IF0VFL	51642	51642		
9C0MPG0	51650	51650		
910LUSA	51743	51743		
8CPXI	51765	51763		
8CPXR	51763	51763		
9STCPX	51767	51767		
9RT0I	52001	52001		
9ID0SET	52056	52056		
9SFTUPV	52111	52111		
9SETUPN	52156	52156		
9SETUP1	52244	52244		
0VERFL	52263	52263		
DVCHK	52263	52263		
9ATAN2	52371	52302		
9SGRT	52513	52513		
8ADDRRLQU	53374	52657		
8RLQUEUE	53373	52657		
7RLINIT	53244	52657		
9R0DDATA	52755	52657		
9R0RDATA	52657	52657		
8TX3	53546	53442		
8TX2	53545	53442		
8TX1	53544	53442		

\*\*\*\* MEMORY MAP \*\*\*\*

LABEL	ENTRY	ORIGIN	SIZE	ERRORS
8MDWRD	53543	53442		
8ENDIOL	53542	53442		
8ARGADR	53541	53442		
9ENDIOL	53503	53442		
9IBDATUM	53451	53442		
9DATA	53442	53442		
9GETBUFF	53553	53553		
8EDITFLG	60341	54557		
7EDITPB	57600	54557		
7EDITIB	57356	54557		
9BFDIT	54566	54557		
9IEDIT	54557	54557		
9DECODE	60431	60425		
9INPUT	60611	60611		
9BCDWRIT	61703	61703		
9BCDREAD	61763	61763		
M\SWAPS	00407	00000		
M\BVFLFL	00406	00000		
M\NAME	00401	00000		
M\TYPE	00313	00000		
M\PRINT	00275	00000		
M\EXIT	00264	00000		
M\FRR	00257	00000		
M\LOCK	00246	00000		
M\FREE	00243	00000		
M\CHEK	00236	00000		
M\DBIR	00227	00000		
M\OPEN	00220	00000		
M\TEST	00215	00000		
RESTART	62165	62061		
TOINVR	62307	62317		
TRANSP	63225	63225		
D\CDPRD	63325	63325		
MSUM	63443	63443		
MVPRD	63547	63547		
MPRD	63665	63665		
VSUM	64056	64056		
V\PRD	64142	64142		
VIPRD	64252	64252		
TRADEOFF	64326	64326		
SEVAL	65357	65357		

\*\*\*\* PROGRAM LIMITS \*\*\*\*

PROGRAM LOWER BOUND ■ 51174  
PROGRAM UPPER BOUND ■ 77777

COMMON LOWER BOUND ■ 12520  
COMMON UPPER BOUND ■ 45432

STARTING LBC 66123

INPUT

CARD NOT USED 298

•END-OF-FILE•





TE 662 .A3  
123  
U.S. FEDERAL  
ADMINISTR  
REPORT NO.

Form DOT F 1720.  
FORMERLY FORM DOT

## **FEDERALLY COORDINATED PROGRAM OF HIGHWAY RESEARCH AND DEVELOPMENT (FCP)**

The Offices of Research and Development of the Federal Highway Administration are responsible for a broad program of research with resources including its own staff, contract programs, and a Federal-Aid program which is conducted by or through the State highway departments and which also finances the National Cooperative Highway Research Program managed by the Transportation Research Board. The Federally Coordinated Program of Highway Research and Development (FCP) is a carefully selected group of projects aimed at urgent, national problems, which concentrates these resources on these problems to obtain timely solutions. Virtually all of the available funds and staff resources are a part of the FCP, together with as much of the Federal-aid research funds of the States and the NCHRP resources as the States agree to devote to these projects.\*

### ***FCP Category Descriptions***

#### **1. Improved Highway Design and Operation for Safety**

Safety R&D addresses problems connected with the responsibilities of the Federal Highway Administration under the Highway Safety Act and includes investigation of appropriate design standards, roadside hardware, signing, and physical and scientific data for the formulation of improved safety regulations.

#### **2. Reduction of Traffic Congestion and Improved Operational Efficiency**

Traffic R&D is concerned with increasing the operational efficiency of existing highways by advancing technology, by improving designs for existing as well as new facilities, and by keeping the demand-capacity relationship in better balance through traffic management techniques such as bus and carpool preferential treatment, motorist information, and rerouting of traffic.

#### **3. Environmental Considerations in Highway Design, Location, Construction, and Operation**

Environmental R&D is directed toward identifying and evaluating highway elements which affect the quality of the human environment. The ultimate goals are reduction of adverse highway and traffic impacts, and protection and enhancement of the environment.

#### **4. Improved Materials Utilization and Durability**

Materials R&D is concerned with expanding the knowledge of materials properties and technology to fully utilize available naturally occurring materials, to develop extender or substitute materials for materials in short supply, and to devise procedures for converting industrial and other wastes into useful highway products. These activities are all directed toward the common goals of lowering the cost of highway construction and extending the period of maintenance-free operation.

#### **5. Improved Design to Reduce Costs, Extend Life Expectancy, and Insure Structural Safety**

Structural R&D is concerned with furthering the latest technological advances in structural designs, fabrication processes, and construction techniques, to provide safe, efficient highways at reasonable cost.

#### **6. Prototype Development and Implementation of Research**

This category is concerned with developing and transferring research and technology into practice, or, as it has been commonly identified, "technology transfer."

#### **7. Improved Technology for Highway Maintenance**

Maintenance R&D objectives include the development and application of new technology to improve management, to augment the utilization of resources, and to increase operational efficiency and safety in the maintenance of highway facilities.

\* The complete 7-volume official statement of the FCP is available from the National Technical Information Service (NTIS), Springfield, Virginia 22161 (Order No. PB 242057, price \$45 postpaid). Single copies of the introductory volume are obtainable without charge from Program Analysis (HRD-2), Offices of Research and Development, Federal Highway Administration, Washington, D.C. 20590.

DOT LIBRARY



00055720

